A Pen-based Paperless Environment for Annotating and Marking Student Assignments

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

Paul Mason
Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand
prmason@slingshot.co.nz

Abstract

A paperless environment for annotating student assignments is appealing to teachers and students. However, to do this, while retaining the richness and ease of annotating the work with a red pen, has not been possible until recently.

This project presents an annotation problem that requires digital annotation, and additionally functionality to properly support the user requirements to move efficiently between assignments, and simultaneously annotate and record marks for the assignment.

With Penmarked, our prototype system, the teacher is able to annotate a digital document with a stylus and at the same time write scores which are recognized and saved into both the student's document and a standard file format. The evaluation suggests that Penmarked is a viable alternative to both paper and existing paperless environments.

Keywords: Annotation, pen-based interaction, paperless environment, online marking.

1. Introduction

Providing feedback to students on their work is an important part of the learning/teaching cycle (Price and Petre 1997). Red-ink annotation onto a paper copy of the assignment is the traditional solution. More recently electronic transfer and paperless marking of assignments has become common practice. This offers advantages to both the student and the marker. A paperless environment allows submission and marking to be carried out: without regard to geographic location; at a lower risk of lost work; with elimination of paper printing time, paper usage and the overheads to support the physical collection and distribution of paper (Price and Petre 1997; Joy and Luck 1998; Preston and Shackelford 1999).

Copyright © 2006, Australian Computer Society, Inc. This paper appeared at the Seventh Australasian User Interface Conference (AUIC2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 50. Wayne Piekarski, Ed.

Reproduction for academic, not-for profit purposes permitted provided this text is included.

Our goal with this research is to explore the requirements for a paperless environment to provide both faculty and students with 'the best of both worlds'. This is to say, an electronic environment that provides rapid, easy storage and transmission of documents, and an informal paper-like environment that affords quick commenting, rich feedback and the recording of scores.

There a many electronic environments for marking assignments, however, providing adequate feedback to the student is more difficult with a paperless environment. Current technology offers three alternatives: alteration of the original document by inserting comments; ink-over, using a tool such as Adobe Acrobat™ and OneNote™, or comments in a separate document. Neither inserted comments nor ink-over include mechanisms for recording and recognising scores or support rapid transition from assignment-to-assignment.

On a tablet PC the user can write directly on the display surface to create digital ink. With this interface and appropriate software support there is the potential for the marker to annotate a script with comments and scores. The scores can be parsed by a recognition engine and recorded in a database. The annotated document and scores can then be returned electronically to the student. Thus the expressiveness of pen annotation, with its associated benefits, is maintained in a paperless environment.

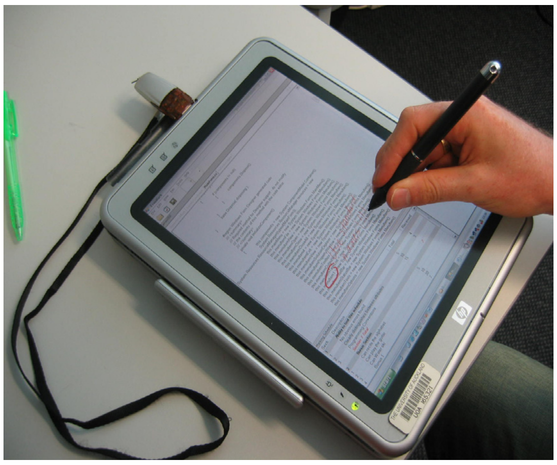


Fig. 1. Tablet Interface for Penmarked

We describe here an environment that could be used for any digital documents. Our prototype includes additional features specific to annotating computer program code. Program code differs from essays and reports in: structure, multi-file persistence and the executable nature of the product. Furthermore, programs are usually compiled and executed as part of the marking process. Others have developed methods to electronically test and evaluate programs. We envisage that these tools could be used along side this environment.

This problem is situated clearly in the education environment where providing effective feedback to students, grading and the ability to handle multiple documents quickly are important. However, much of what we have learnt, and describe here, is applicable to more general document review and annotation problems.

The structure of the remainder of this paper is as follows: the next section provides a background for the work. Then we describe the design requirements and implementation of the prototype, Penmarked and its evaluation. This is followed by a summary of related work on pen computing, annotation and paperless marking and a discussion. Finally, we draw conclusions and suggest further areas of research.

2. Background

This work was inspired by our desire to have an efficient electronic environment to mark student assignments without compromising in the area of feedback. First, let us consider three alternate ways of providing feedback on digital documents: ink-over, inline comments inserted into the document such as is available with Microsoft Word ™ Review, and offline comments in a separate document.

Ink-over is the richest type of annotation. It allows the reviewer to indicate position by underlying, circling, pointing or highlighting and add comments with words, pictures or pictograms. There is no restriction as to the nature of the annotation and, as the original document is unchanged, the feeling of ownership of the document remains with the author (Sellen and Harper 2002). Inline comments, such as are provided by Microsoft Word, allow the reviewer to change the content of the document, while this is very effective for co-authoring activities it is inappropriate when reviewing a completed document. Offline comments require the reviewer to reference the point in the original document and then make their comment. Typically these are typed and restricted to text. They are slow to create because of the need to define a point of reference and lack the expressiveness of *red ink.*

The visual picture of an annotated document conveys meaning. With ink-over, because the layout of the original document is retained, the nature of the annotations is obvious. It is clear when the annotations are predominately syntactic corrections or when the marker has put a big tick, cross or question mark beside a section. In contrast, inline comments disrupt the visual image of the original without indicating the types of changes suggested. Offline comments do not disrupt the original document but because of the overhead of providing this type of feedback the comments are often minimal or vague.

Besides annotating an assignment the marker is required to score the work, typically assignments are judged on a number of factors; for example, content, presentation and references. The annotations act as a prop to the marker when deciding on the score for each factor, here again the visual image provided by ink-over is easier to survey that either inline of offline comments. The scores for each factor are then totalled. These scores are returned to the student, and recorded in a database for ranking and grading.

Existing tools that support ink-over and in-line comments require double recording of scores (on the document and into a database). Offline comments can be supported in a separate digital document that also includes attributes for recording scores. In this case, each value is written once only, with totalling and archiving into a database being trivial.

A last, but important, consideration with marking software is support for work practises. Markers need to be able to gather and return work simply, and move between assignments easily and quickly. Computer programming assignments present some particular additional requirements; program code is often held in more than one file and usually examined in a non-sequential manner, therefore all the parts of the program need to be available to the marker at one time. Additionally markers usually compile and execute the program as a part of the marking process.

3. Design

A tool for paperless annotating and grading of student assignments requires support for three major functions: annotation of the script, recording scores and switching between assignments. Included in this section are some specific requirements for marking computer programs as this is our focus of interest.

3.1. Annotation

The marker must be able to ink anywhere on the script, much as they would a piece of paper. The design strategy is simple, unrestricted writing and erasing equivalent to pencil and paper. Other functionality such as a clipboard of comments, may be useful (Price and Petre 1997), however it is not clear that markers, do, or should, repeat the exact same comment.

To support multi-file program code an interface similar to many integrated development environments (IDEs) with each file displayed on a separate tab is proposed. This allows the marker to move rapidly between the files.

Reflowing ink on digital documents is challenging (Brush, Bargeron et al. 2001). However, this is only required if the underlying document is reformatted, assignments can be regarded as fixed except for pagination for printing. Rather than complex analysis of annotations our approach is to indicate on the annotation frame where page-breaks fall so that the marker is alerted to annotations that will cross a page boundary. Also, any annotation that crosses a boundary can be partially duplicated in the page margins.

3.2. Scores

Given the idiosyncratic nature of annotations (Marshall 1997), it is not possible to recognize scores anywhere on an annotated page. Yet pen input is preferable as it is distracting to move between pen and keyboard (Jarrett and Su 2003). Three alternative ways to indicate which ink should be recognized as scores were considered; (a) the user could change inking mode and write scores directly onto the document, (b) a score area in a specific region of the page (column or box), (c) a separate section of the interface to record scores. Option (a), ink modes, was discarded because experiments with modal inking have shown that it is confusing and error prone (Plimmer and Apperley 2003). A score region (b) would be suitable for some assignments, but in some cases scores are not allocated for a particular portion of the work but for an overall characteristic (for example presentation), in this situation a summary table (c) of criteria and scores is more appropriate. Regardless of the strategy adopted, the scores need to be recognized, totalled and saved in a standard format for use in student records.

There are a wide variety of scoring mechanisms used such as numeric values, Likert scales and achieved/ not achieved. Provision of different scoring mechanisms do not present difficult challenges from a software implementation perspective. We envisage a number of the more common mechanisms should be provided and that the user can customize this part of the tool.

3.3. Work Practises

Efficiency is important. Marking is time intensive and therefore expensive. Moving to the next assignment needs to be as easy as picking up the next paper from a stack. As with annotation, people employ diverse practices in the order that they mark assignments (Price and Petre 1997; Preston and Shackelford 1999) (for example strip marking or whole script), therefore a way to indicate the status of an assignment, such as not opened, started, and completed, is necessary.

Computer programs are usually compiled and run as a part of the marking process; given the variety of program languages specific functionality for this, in a general annotation tool, is impractical; rather the design goal is to make it easy to be able to move between the integrated development environment (IDE) or automatic testing tool and marking environment.

Assignments are often submitted as a zipped package. With multi-file programs a standard technique is for students to zip all the files together and place the zipped file into an electronic drop-box. The marker collects the assignments from the drop-box, unzips the package, compiles and executes the program and writes feedback to the student while also grading the work.

Using standard tools, at least eight taps/clicks are required to unzip an assignment package and open it in both the (IDE) and a marking system. Part of this process needs to be repeated each time an assignment is opened (Price and Petre 1997). This will not permit markers to quickly turn to another assignment. A design goal is to reduce this to one or two clicks/taps to move between assignments.

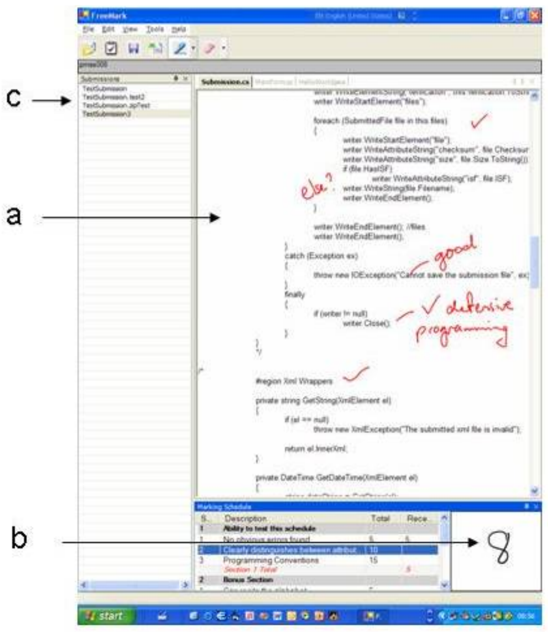
The proposal is that a part of the main interface lists all assignments and indicates the current status of each. A tap on an assignment in the list opens the assignment for marking and prepares the submitted files for compiling and testing.

When marking is completed the annotated assignment along with the marking schedule needs to be saved in a standard format so that it can be returned to the student. The final requirement is for scores to be available for student management purposes.

There are a wide variety of student management systems used for collecting and returning work to students. A web-service is an accepted approach to interface to these systems.

4. Implementation

The Penmarked prototype is describe in detail in this section. The software affords digital annotation, scoring and support for work practises. It utilizes tablet PCs and the Windows XP Tablet™ operating system.



```
                writer.WriteStartElement("files");

                foreach (SubmittedFile file in this.files)
                {
                        writer.WriteStartElement("file");
                        writer.WriteAttributeString("checksum", file.Checksum
                        writer.WriteAttributeString("size", file.Size.ToString())
                        if (file.HasISF)
                                writer.WriteAttributeString("isf", file.ISF);
                        writer.WriteString(file.Filename);
                        writer.WriteEndElement();
                }

                writer.WriteEndElement(); //files
                writer.WriteEndElement();
        }
        catch (Exception ex)
        {
                throw new IOException("Cannot save the submission file", ex);
        }
        finally
        {
                if (writer != null)
                        writer.Close();
        }
}
*/

#region Xml Wrappers

private string GetString(XmlElement el)
{
        if (el == null)
                throw new XmlException("The submitted xml file is invalid");

        return el.InnerXml;
}

private DateTime GetDateTime(XmlElement el)
{
```

S..	Description	Total	Rece..
1	Ability to test this schedule		
1	No obvious errors found	5	5
2	Clearly distinguishes between attribut	10	
3	Programming Conventions	15	
	Section 1 Total		5
2	Bonus Section		

Fig. 2. Penmarked Standard Interface Layout including (a) annotation pane, (b) mark schedule, (c) student list.

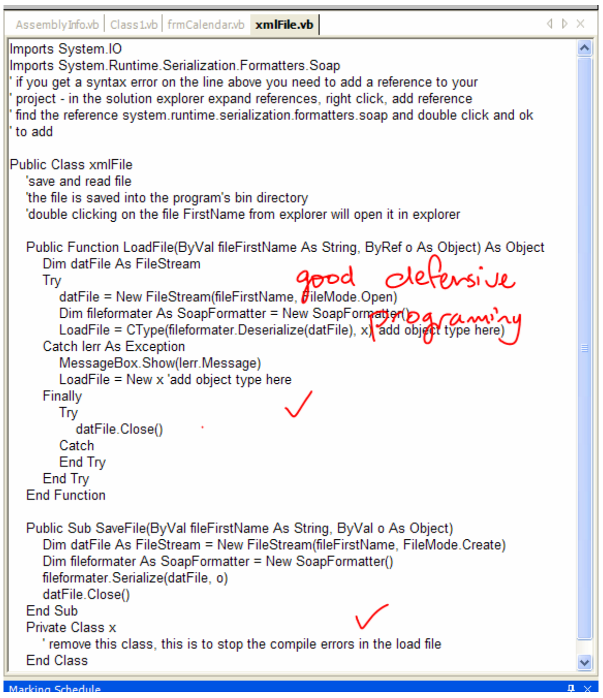
Tablet PCs provide a convenient hardware interface where the user can ink directly onto the output screen. XP Tablet includes digital ink as a native data type and character recognition is integrated into the operating system. The ink API provides basic support to the programmer. Penmarked is written in C#, using the Microsoft .Net framework.

Figure 2 shows the standard layout of the Penmarked user interface for right-handed users. Standard menus are provided at the top of the window but the most frequent tasks can be accessed via the icon bar. The identifier for the current assignment is also shown at the top of the window. Both the mark schedule and student list panes are resizable and dockable; they can float over the annotation pane or be docked on any perimeter. They can also be docked into the same space, in which case tabs are used to navigate between them. This provides maximum flexibility for right/left handedness and tablet orientation.

The major components of the user interface are: the annotation pane (figure 3), marking schedule (figure 5), and student list (figure 4).

4.1 Annotations

The annotation pane (figure 3) consists of a multi-tab space. Each selected file from the current assignment is displayed in a tab. The tabs are titled with the file name; typically this is also the program class name. The user can switch between files by tapping the tabs, and navigate down a file with the scroll bars.



```
Imports System.IO
Imports System.Runtime.Serialization.Formatters.Soap
' if you get a syntax error on the line above you need to add a reference to your
' project - in the solution explorer expand references, right click, add reference
' find the reference system.runtime.serialization.formatters.soap and double click and ok
' to add

Public Class xmlFile
   'save and read file
   'the file is saved into the program's bin directory
   'double clicking on the file FirstName from explorer will open it in explorer

   Public Function LoadFile(ByVal fileFirstName As String, ByRef o As Object) As Object
      Dim datFile As FileStream
      Try
         datFile = New FileStream(fileFirstName, FileMode.Open)
         Dim fileformater As SoapFormatter = New SoapFormatter()
         LoadFile = CType(fileformater.Deserialize(datFile), x) 'add object type here)
      Catch lerr As Exception
         MessageBox.Show(lerr.Message)
         LoadFile = New x 'add object type here
      Finally
         Try
            datFile.Close()
         Catch
         End Try
      End Try
   End Function

   Public Sub SaveFile(ByVal fileFirstName As String, ByVal o As Object)
      Dim datFile As FileStream = New FileStream(fileFirstName, FileMode.Create)
      Dim fileformater As SoapFormatter = New SoapFormatter()
      fileformater.Serialize(datFile, o)
      datFile.Close()
   End Sub
   Private Class x
      ' remove this class, this is to stop the compile errors in the load file
   End Class
```

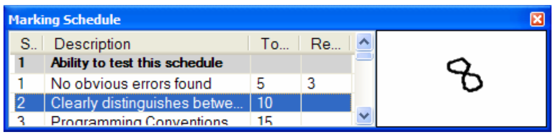
Fig. 3 Annotation Pane

The marker can ink over the document freely. They are also able to erase the ink annotations. The width of the annotation page is fixed to fit standard A4 paper. Lines of text longer than the available space are automatically wrapped as the file is loaded. Markers cannot change the text in the student's document.

Inking is achieved by placing the ink on a transparent overlay. Although the .NET API provides most of what is required there are a number of event responses where direct access to the Windows API is required to synchronize the ink and text as the windows scroll.

4.2 Scoring

The marking schedule (Fig. 4) is displayed as a table in a separate pane. The table consists of sections and subsections. Each item has a minimum and maximum value. In order to provide adequate space for writing and minimize the space required for this pane, the score is written in a box on the right-hand side of the pane. When the marker taps another item in the schedule or after a customizable time-lapse, the mark is automatically recognized and entered into the currently selected item.



S..	Description	To...	Re...
1	**Ability to test this schedule**		
1	No obvious errors found	5	3
2	Clearly distinguishes betwe...	10	
3	Programming Conventions	15	

Fig. 4 Marking Schedule

Recognition success rates are optimized by using the factoid feature in the tablet recognizer. This restricts the recognition to digits and arithmetic signs. The input is also validated against the maximum and minimum. If the input is valid the value is saved into the schedule and the writing box flashes green. Otherwise the writing box flashes red and a large message is displayed over the marking schedule, no value is recorded. This is fallible, to minimize the risk, before creating an output document the system checks if there are any missing values and alerts the user if empty cells are found. Subsection totals and a grand total are calculated each time a value is added to the schedule.

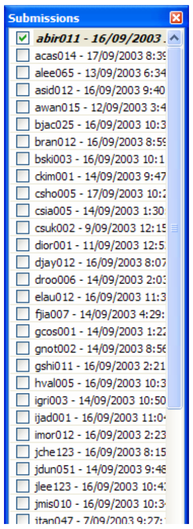
The marking schedule can be edited by the teacher at the beginning of a marking session and is saved for reuse as an XML file. We chose to implement numeric scores in this prototype as this is the most general solution. Other scoring mechanisms such as Likert scales have not been implemented, but we do not foresee any difficulty with this.

4.3 Support for Work Practise

Work practise support minimizes the user effort to collect and distribute assignments and transition between assignments. We have implemented a simple interface to our department electronic drop-box and provide add-in access to support different drop-box implementations. These services collect the assignments and associated student information, unpackage the data into a standard windows folder structure and create a student submission file for each student.

After this, to collect submissions Penmarked parses the folder structure looking for submission files and lists them in the student list pane (figure 4). These submission files are small XML files that accompany each assignment and hold information such as student name, id, email address and the names of the assignment files and marking status.

When marking essays and other non-functional assignments the user simply taps on an entry in the list to open the assignment. He/she can then scroll down the document, reading and annotating in the same way that they would with the paper equivalent. Scores can be added to the mark schedule at any point.



☑ *abir011 - 16/09/2003*
☐ acas014 - 17/09/2003 8:3
☐ alee065 - 13/09/2003 6:34
☐ asid012 - 16/09/2003 9:40
☐ awan015 - 12/09/2003 3:4
☐ bjac025 - 16/09/2003 10:3
☐ bran012 - 16/09/2003 8:5
☐ bski003 - 16/09/2003 10:1
☐ ckim001 - 14/09/2003 9:47
☐ csho005 - 17/09/2003 10:
☐ csia005 - 14/09/2003 1:30
☐ csuk002 - 9/09/2003 12:1
☐ dior001 - 11/09/2003 12:5
☐ djay012 - 16/09/2003 8:07
☐ droo006 - 14/09/2003 2:0
☐ elau012 - 16/09/2003 11:3
☐ fjia007 - 14/09/2003 4:29:
☐ gcos001 - 14/09/2003 1:2
☐ gnot002 - 14/09/2003 8:5
☐ gshi011 - 16/09/2003 2:21
☐ hval005 - 16/09/2003 10:3
☐ igri003 - 14/09/2003 10:50
☐ ijad001 - 16/09/2003 11:0
☐ imor012 - 16/09/2003 2:23
☐ jche123 - 16/09/2003 8:15
☐ jdun051 - 14/09/2003 9:4
☐ jlee123 - 16/09/2003 10:4
☐ jmis010 - 16/09/2003 10:3
☐ jtan047 - 7/09/2003 9:27:

Fig. 5 List Pane

Marking programs may involve extra steps. The opening process may include unzipping file packages and selecting appropriate files for display. If the assignment includes a zip file, Penmarked automatically extracts the files from the package and places them in a subdirectory of the directory that contains the students work. As programming projects often include files that the marker does not look at, the files are searched for particular file by type or name (eg "*.vb"). Also particular files can be excluded, for example VB .Net projects include an 'assemblyInfo.vb' which is unlikely to be of interest to the marker although it has the same file extension as the code files. Each opened file is placed in its own tab in the annotation pane.

To run the program the user taps an icon on the main Penmarked icon bar to go to the student's assignment folder which has been automatically opened on the desktop. From this folder the marker can run the .exe file or start the IDE. The marker can then move freely between Penmarked, where they can write feedback on the assignment and record scores, and the IDE or running program. Where automatic electronic marking is undertaken the reports from this can be included as one of the files available in the Penmarked interface or independently in the student's assignment folder.

When the marking of a submission is completed, an ink file is saved with the assignment files and the output PDF file is saved and emailed to the student. The scores from all submissions are saved in an XML file from which they can be later imported into any standard database or spreadsheet.

4.4 Output File

When marking is finished a PDF file is generated for return to the student. This consists of a title, the filled-in marking schedule and the annotated document. The files in the annotation pane are not paginated; therefore ink annotations may flow across page boundaries. Any ink which is identify as crossing a page boundary is duplicated, as much as possible, in both the bottom margin of the first page and top margin of the following page.

5. Evaluation

Both ongoing informal evaluations and a formal evaluation study have been completed. During development ongoing informal evaluations were undertaken. Experienced markers were observed marking 2 or 3 assignments, and asked to comment and make suggestions. From these studies a number of changes were made such as increasing the sophistication of the files selected for display and adding a 'find' function to the annotation pane. Subsequently a field-trial was undertaken. As this is a novel application, in a new interaction paradigm a two pronged approach was taken: think-aloud and a focus group.

5.1 Think-Aloud

The think-aloud study followed a standard protocol requiring the participants to verbalize what they are doing and thinking whilst working with the software. To accurately record their expressions, body language, opinions and actions, the users were video taped and observation notes were kept.

Four markers participated in this study. They were unfamiliar with the Tablet PC and the Penmarked program. They had, however, used a variety of other marking support systems that are in use in the department. Most of these support offline commenting and a marking schedule, with comments and scores returned to the student by email. Each marker marked about 30 assignments. They were observed and asked to think-aloud for the first three and final three assignments.

Persuading the markers to talk was extremely difficult; much worse than anticipated. They appeared to be concentrating on the programs and seemed to find talking a distraction. Often when they did speak, it was about the program that they were marking, not Penmarked. This was attributed to the cognitive demands of programming.

This study uncovered a number of interesting issues. Toolbar icons were used for most tasks; however during system start up some markers used the menus to open up the submission folder. This created difficulties due to the participants' inexperience in use of the pen. They often selected an unintended menu item. A start-up wizard may be the solution.

Our design focus on efficient work practises support was reinforced by a 'bug'. When closing a submission, a dialog box appeared asking whether to save the work regardless of whether any changes had been made since the last save. All markers commented that this was irritating; clearly a more intelligent save is required!

A number of other small interaction problems were uncovered such as the need for a horizontal scroll bar on the annotation pane and the position of the period when writing a value ('1.5' is correctly recognized while '1·5' is not recognized). Two markers suggested a programming specific feature, colour syntaxing of the code, as it makes it easier to read.

Penmarked has been carefully designed so that keyboard input is not required but the programs that were being marked were not designed for the tablet. Text-entry to the assignments via the on-screen keyboard proved to be tedious, so an external keyboard was added during the first participant's session. After the addition of the keyboard, the pen was used to navigate and to enter annotations, whilst the keyboard was used to input data into the students' assignment and two participants used it to enter scores. We noted that markers used the pen and the keyboard simultaneously.

5.2 Focus Group

The same group of markers were brought together in a focus group after marking was finished. The discussion centred on suggestions for improvements and problems encountered. Particular questions were put to the group on: interface navigation; the tablet hardware; software bugs; general difficulties; general likes. Many interesting comments were made during the forty-five minutes the group spent together.

The users found the interface easy to navigate, and the icons and menus meaningful. The docking windows, editing controls, and automatic zip extraction were all mentioned as making Penmarked easy to use.

The student list pane received favourable comments. First, having a checkbox state beside each assignment meant that the markers could quickly scan the list and know which assignments had been marked and how many remained. Checking-off also eliminated accidentally missing assignments, a problem with other systems they had used. Another problem they had encountered in other systems was assigning scores to the wrong student. They all agreed that the risk of missing a student or assigning the scores to the wrong student was greatly reduced in Penmarked because of the work practice support. They suggested a third state for each assignment, 'in progress' would be useful. In summary the student list and ability to directly move to the assignment folder made marking a much quicker and easier process as compared to other systems. They stated that in this respect Penmarked was the best system that they had used.

They suggested two ways to further streamline workflow. First, to automatically find, open and close the IDE for the marker. Second, associate the marking schedule with a submissions folder. This would allow the marking schedule to be automatically loaded when the submission folder is specified.

They commented that pen entry of scores required accurate writing. One marker had problems entering negative numbers and another with the digit 5. We found in both cases this was a matter of writing style. The vertical position of the '-' is critical for successful recognition. The problem with the '5' was due to the way the marker formed the character (his first written language is Chinese). Having the option for both writing and typing the scores was useful; these two users primarily used the keyboard while the other two used the pen. Others have suggested providing an on-screen numeric keypad as an alternative to the score inking box; this would alleviate interaction problems for people who have difficulty with the recognizer.

The Tablet PC hardware was deemed exceptional. However, because the programs that were being marked were not Tablet applications, having an external keyboard was an aid to marking. An interesting observation made by one of the markers was that they had initially tried erasing annotations with the back of the pen. While this is supported with some Tablet PC's, it is not with the hardware that was used for this study. The mandatory save dialog box on close noted in the think-aloud study, was raised again.

6. Related Work

This section describes related work in pen interfaces, annotation and paperless marking software. Pen interfaces provide a very different interaction paradigm to keyboard and mouse. The interaction design drew on the experiences of Plimmer and Apperley (2003; 2003) using digital whiteboards and Jarrett and Su's (2003) suggestions on user interface design for tablet applications.

Recognition of digital ink is challenging, particularly if there is a desire for a modeless interface (Plimmer and Apperley 2003). The Microsoft XP Tablet operating system (Microsoft 2002) provides recognition of characters. The factoids feature of the recognizer allows the application programmer to further limit the recognition space.

Digital ink annotations, to retain their meaning, are position-specific; the underlining or circling of a phrase must stay with that phrase for the meaning to be retained. This requires the ink to be reflowed as the document moves. The work of Brush et al. (2001) and

Golovchinsky & Denoue (2002) suggest approaches to this problem.

Research into ink annotation has focused on the annotation requirements for digital books. Marshall (1997), carried out an extensive survey on the annotations in second-hand textbooks, she found that "annotations are informal, ad hoc and take many forms". A further study (Shipman, Price et al. 2003) confirmed the idiosyncratic nature of annotations. A number of software tools have been developed in related areas. For example the XLibris project (Schilit, Golovchinsky et al. 1998) explored the use of a tablet computer to support active reading of digital books.

Wolfe (2000), compared the effect of an annotated versus non-annotated article on how students read and wrote about the article. She concluded that: in general terms the more annotation the better the students did at locating points of interest in an article.

Studies that have examined the annotations of markers (Price and Petre 1997; 2004) have found that these vary as widely as those of readers' annotations. Additionally, Heinrich and Lawn (2004) noted, scoring practises varied widely; for example some markers use ticks and crosses while others write values (e.g. 1 ½).

Paperless environments for marking programming assignments are not new. For example (Joy and Luck 1998) describe a system which supports electronic submission, testing and evaluation which can be supplemented with manual feedback via off-line comments.

Price and Petre (1997) compared the nature, form and quality of feedback on programming assignments between electronic feedback using Microsoft Word™ review functionality and pen and paper. They noted that one of their three markers showed evidence of his/her style being constrained by the text features of the word processor until, in later assignments, he/she discovered the freehand drawing tool. Another of the markers in this study strip marked questions. This marker encountered problems with the overheads involved with opening and switching between documents. They observed less use of emphasis with the electronic marking but higher legibility of the typed text.

Preston and Shackelford (1999) have also looked specifically at marking of program code. They suggest that being able to view the work at different levels of abstraction is important.

7. Discussion

This section comments on: the design strategies; implementation of the prototype; and the evaluation studies. The initial design criteria are examined in light of the implementation and user experience and possible enhancements and changes of direction. Also, the findings of the two evaluation studies are reviewed.

Our design goal was to provide a paperless environment for marking student assignments that incorporated the ability to annotate work in an unconstrained manner with a pen. This has been achieved. Our analysis lead to the software having three major functional areas: annotation panel, marking schedule and student list.

The annotation panel, while technically the most difficult to implement, worked well in the trial, with exception of the lack of an end-of-pen eraser it received only positive comments from the users.

The marking schedule section also worked well for the particular assignments marked. In practice, a wide variety of different scoring methods are used – from Likert scales to simple single letter grades – implementation of a range of different scoring methods could be included into Penmarked. Also the users' suggestion of automatically loading a schedule with the assignments would increase work efficiency.

The high priority placed on providing an efficient environment was both recognized and appreciated by the users. The fact that the software did not know when work did not need saving, thus requiring the users to save unnecessarily, so annoyed the users is seen as evidence of the importance of a streamlined workflow for this application.

Penmarked is specifically designed to support 'pen only' interaction. Penmarked is indeed very useable in this mode. The programs that were marked during the case study required keyboard input; it quickly became apparent that the on-screen keyboard was too slow. The best environment, in this particular case, was the tablet with a keyboard attached. This reinforces the view that interfaces must be specifically designed for pen interaction (Jarrett and Su 2003).

A later informal experiment used two computers connected to a shared network space; the tablet computer ran Penmarked and another computer ran the student program in the IDE. While hardware intensive, this appeared to provide a better workspace as both the running program and annotation copy can be viewed simultaneously. Clearly this is specific to marking work that has a functional computer-based component such as a program and would not be necessary for marking more standard assignments such as essays.

The evaluation techniques used provided both detailed and high level feedback on the usability of the software. The think-aloud experiment supplied comprehensive information on interaction and program bugs. In contrast, the focus group provided a more conceptual review of the system. The participants commented that they found it extremely difficult to be filmed, talk and concentrate simultaneously and found the filming intimidating. Yet in the focus group they were comfortable talking, and conveying their opinions and feelings.

Faculty use other automated techniques for marking programs for example automatic testing and program metrics. We do not see Penmarked as replacing these tools, rather as a general assignment annotation and grading tool that, in the case of programs, could be used along side other tools.

8. Conclusions and Future Work

This work has explored the requirements of a paperless environment for the marking of assignments where the marker can annotate the work as if they were working with paper. The design goal was to produce an efficient easy-to-use environment. The implementation has shown that this is technically possible and the evaluation studies suggest that the goal of an efficient easy-to-use environment was achieved. Our evaluation study was conducted on programming assignments; from an interaction perspective these are more complex to mark on a computer than essays or reports. Penmarked has subsequently been successfully used for essay marking.

Penmarked supports rich-text and text files; our current work is focused on supporting a wider range of document formats such as word processor and PDF documents. There are many features that could be added to Penmarked, each needs to be carefully designed and evaluated to ensure that usability is maintained and that the natural interaction paradigm of pen-on-paper is not lost.

One feature which is often suggested is a clipboard of frequently used comments. This is worthy of further investigation and experimentation; are comments repeated verbatim as frequently as markers believe or as one of Price and Petre's (1997) markers commented, is each student worthy of an individualized response?

A study similar to that of Price and Petre's (1997) comparing feedback in Penmarked and other electronic and manual environments would be interesting. Along side this student perceptions of the effectiveness of different feedback mechanisms and support systems could be examined.

9. References

Brush, A. B., D. Bargeron, et al. (2001). Robust annotation positioning in digital documents. Sigchi'01, Seattle, WA, ACM.

Golovchinsky, G. and L. Denoue (2002). Moving markup: repositioning freeform annotations. Symposium on User Interface Software and Technology, Paris, France, ACM.

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

Jarrett, R. and P. Su (2003). Building Tablet PC applications. Redmond, Microsoft Press.

Joy, M. and M. Luck (1998). Effective Electronic Marking for On-Line Assessment. ITiCSE, Dublin, ACM.

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

Microsoft (2002).

Plimmer, B. E. and M. Apperley (2003). Freeform: A Tool for Sketching Form Designs. BHCI, Bath.

Plimmer, B. E. and M. Apperley (2003). Software for Students to Sketch Interface Designs. Interact, Zurich.

Preston, J. A. and R. Shackelford (1999). Improving on-line assessment: an investigation of existing marking methodologies. ITiCSE, Carcow, Poland, ACM.

Price, B. and M. Petre (1997). Teaching programming through paperless assignments: an empirical evaluation of instructor feedback. ITiCSE, Uppsala, Sweden, ACM.

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

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

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

Wolfe, J. L. (2000). Effects of annotations on student readers and writers. Digital Libraries, San Antonio, TX, ACM.