Designing an Environment for Annotating and Grading Student Assignments

Dr Beryl Plimmer and Paul Mason
Department of Computer Science
University of Auckland, New Zealand
Email: {bpli001|pmas008}@ec.auckland.ac.nz

Abstract

A number of software tools are available to annotate documents with digital ink. However, they only partly solve the problem of annotating and grading student assignments, this task requires digital annotation capabilities, recognition of digital ink and support for workflow practices. We are particularly interested in marking student programming assignments. Programs differ from essays and reports in that they: often need to be examined non-sequentially, exist in multiple files and are usually compiled and executed as a part of the marking process. In this paper we describe the design process for our initial design of such a paperless environment, describe the design decisions we have made and our first prototype, 'Penmarked'. We discuss how this work may inform others designing pen-based applications and conclude with future work.

Keywords

Paperless environments, annotation, pen-based computing, design strategies.

INTRODUCTION

The electronic transfer of assignments offers advantages to both students and teachers. In an electronic environment: submissions can be time-stamped and receipted automatically, the documents distributed electronically to multiple markers, scores directly recorded in a database and marked work returned to the student electronically. In addition, a paperless environment allows: work to be carried out without regard to geographic location, reduces the risk of lost work, paper usage and the overheads to support the physical collection and distribution of paper.

However, providing adequate feedback to students is more difficult in a paperless environment. There are three main approaches with current technology: inserting comments into the original document, ink-over and off-line comments. When a marker inserts comments into the original document the layout and flow of the work is disrupted. A number of tools support ink-over (for example Adobe Acrobat) but do not deal elegantly with multi-file program code. Both inserted comments and ink-over tools do not include mechanisms for recording or recognising scores. The third alternative is for the reviewer to write comments and scores in a separate document that can then be sent to the student, however this requires extra effort by the reviewer to reference the point in the original document (e.g. "In page 2, paragraph 3 you …..."), before writing the comment. The advantage of the third approach is that the marks are recorded once in a digital format that can then easily be used for student records.

Our particular interest is in marking computer programs. Program code differs from essays and reports in structure, multi-file persistence and the executable nature of the product. While 'normal' student assignments are designed to be read sequentially, programs conform to quite a different structure based on objects, data and functionality. Programs are not intended to be read as sequential documents so it is often necessary when marking them to jump around following the flow of execution. Furthermore, unlike essays which usually exist as one file, many programming languages such as Java and the .Net languages support saving each class in a separate file. Finally, it is usual when marking programs to compile and execute the code.

Our goal with this research is to design a rich paperless environment that provides both students and markers with 'the best of both worlds'. We describe here the design of 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 Figure 1.

Figure 1 Penmarked on Tablet

The structure of the remainder of this paper is as follows. The next section reviews related work, this is followed by the design proposal. A first prototype is described in the implementation section. After this the discussion section examines our design, development and evaluation process and how this may guide others working in this field. The last section details our future work plans and other research questions that this work raises.

RELATED WORK

Research into ink annotation of documents has mostly 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". The types of annotations that she observed fall roughly into two categories; selection marks (underlining, circling etc) and interpretive marks (comments and glyphs). The annotations that Marshall studied were essentially the reader's private annotations. A further study (Shipman et al., 2003) confirmed the idiosyncratic nature of annotations. Heinrich and Lawn (2004), examined the annotations of markers and found a similar pattern of comments, additionally scoring practises varied widely; for example some markers use ticks and crosses while others write values (e.g. 1 ½).

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, the positive and negative annotation comments had an effect on the students rating of the persuasiveness of particular parts of the article, and the content of essays the students wrote from the material differed depending on whether the article was annotated and the type of annotation.

A number of software tools have been developed in related areas. For example the XLibris project (Schilit et al., 1998) explored the use of an early tablet computer to support active reading of digital books. Heinrich and others (2004; 2003) have proposed text based annotation of student assignments. Our own work with digital whiteboards (Plimmer & Apperley, 2003) together with those of Jarrett and Su (2003) on Tablet applications suggest successful pen interaction techniques; for example moving between pen and keyboard is distracting and should be avoided. The positioning and reflowing of ink annotations is a difficult problem particularly if reformatting of the document is necessary; the work of Brush et al (2001) and Golovchinsky & Denoue (2002) suggest approaches to this problem.

DESIGN

A tool that supports paperless annotating and grading of student assignments requires three major functions: annotation of the script, recording scores and workflow support. 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. Handwriting annotations is quicker than typing and placing widgets with a mouse (Plimmer & Apperley, 2003) and conveys personalisation that is not evident with computer drawn annotations. We propose that the marker should be able to ink anywhere on an assignment, much as they would a piece of paper.

Given the idiosyncratic nature of annotations, it is not possible to recognise marks in this free-format. We considered three possible design alternatives for indicating which ink should be recognised as marks; (a) the user could change inking mode to write marks directly on to the document, (b) a mark column on one side of the page, (c) a separate section of the interface to record marks. We discarded option (a), ink modes, because our previous experience with modal inking showed that it is confusing and error prone (Plimmer & Apperley, 2003). A mark column would be suitable for some assignments, but in many cases marks are not allocated for a particular portion of the work but for an overall characteristic (for example presentation), in this situation a summary table of criteria and marks is more appropriate. In either case the marks need to be recognised, totalled and saved in a standard format for use in student records.

To support multi-file programs we propose an interface similar to many integrated development environments (IDEs) with each code file opened and displayed on a separate tab. It is also usual to compile and run a program as a part of the marking process; given the variety of program languages specific functionality for this is impractical, rather our design goal is to make it easy to be able to move to and from the appropriate IDE to perform this task.

We recognise that marking is usually undertaken in batches and efficiency is important; moving from student-to-student needs to be as easy as picking up the next paper from a stack. As with annotation, people employ diverse practises in the order that they mark assignments (from strip marking section-by-section to marking each independently), therefore a way to indicate the status of an assignment, such as: not opened, started, and completed is necessary. The standard 'file open dialogue' will not permit markers to quickly turn to another assignment or show an assignments marking status. We propose a part of the main interface that lists all assignments and indicates the current status of each. A tap on an assignment in the list will open the assignment for marking or reviewing.

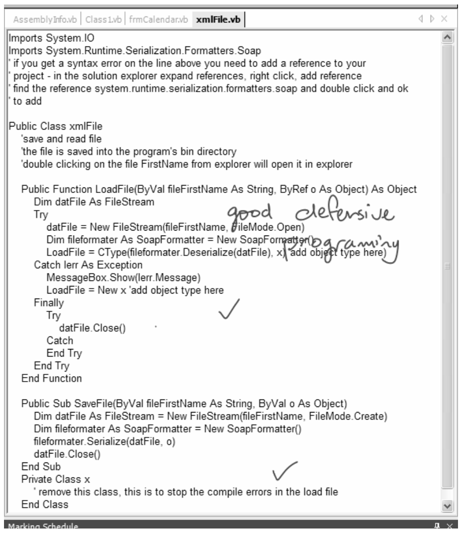
When marking is completed the annotated assignment along with the marking schedule needs to be saved in a standard format so that it can be displayed or printed by the student. The software should also include a mechanism for returning marked assignments to students either by email or placing them in a repository.

Reflowing ink on digital documents is technically challenging, however with assignments we can regard the underlying document as fixed thereby limiting the reflowing to pagination of annotations that fall across a page break. We propose a simplistic approach, retaining the normal flow of program code (unpaginated) while indicating on the annotation frame where page-breaks will 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.

IMPLEMENTATION

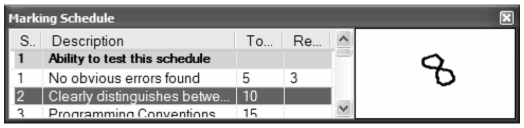
We have implemented an initial prototype for the Tablet PC using C#.Net. The interface has three main frames; assignment display and annotation, mark schedule, and student list. Each frame is a docking window that the user can move or resize. The annotation frame (figure 2) opens a tab for each code file. The paradigm is very much 'write on paper' the user can write anything, anywhere; none of this ink is recognised. Normal editing such as erase, move and resize is supported.

To record marks we have implemented the marking schedule (figure 3). In order to provide enough space to easily write while at the same time preserving screen space the marker writes the score for the currently selected marking criteria in a recognition pane. Recognition occurs either when the marker moves to the next criteria in the marking schedule or after a short time lapse. To increase accuracy of recognition we check that the mark is numeric and in the range of that assessment item. The completed marking schedule is included in the output document and saved into an XML file for student records.



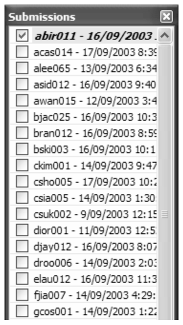
```
Imports System.IO
Imports System.Runtime.Serialization.Formatters.Soap
' if you get a syntax error on the line above you need to add a reference to your
' project - in the solution explorer expand references, right click, add reference
' find the reference system.runtime.serialization.formatters.soap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

Figure 2 Annotation Pane



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	

Figure 3 Marking Pane



- ✓ ***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

Figure 4 List Pane

To provide an interface where the marker can quickly move between assignments, at the beginning of a marking session the program scans a given directory structure for assignments. Each assignment submission is accompanied by a small XML file that includes student identification data, marking status and a list of submitted files. All the assignment submissions with their status are shown in the list frame (figure 4). A double tap on an entry in this list opens the submission; zipped files are unzipped and saved in a subdirectory with the appropriate code files displayed. As the assignments are marked they can be ticked off on the list.

A partially marked submission can be saved at anytime. Fully marked submissions are exported as a PDF file that includes a cover page, the marking schedule and the annotated code. At a later stage these PDF files can be emailed to the student using the email address in the submission file.

DISCUSSION

During development of the prototype described above we undertook three informal evaluations. Experienced markers were asked to mark 2 or 3 assignments using Penmarked while we observed, we also asked them to comment and make suggestions. From these studies we made a number of important changes: the list of assignments to include current status; increased the sophistication of the files selected for display; while, in contrast, found it helpful to unzip the entire project so that the marker could easily compile and run the program; discovered the necessity of a 'find' function; changed the return of marked work to students from immediate on completion of marking to user controlled and included an indication of page-break locations.

With the release of Tablet PCs, pen based annotation and marking software is now a commercial possibility. However the interaction paradigm for tablets is quite different to that of keyboard, mouse & screen. We have presented here some different approaches to interface design that exploit the hardware and recognition characteristics of the Tablet PC. We suggest that many of the standard interactions of the screen/keyboard/mouse environment do not transfer well to a pen-based environment and that it is necessary to carefully rethink design basics. This project has benefited from quick, informal and early usability evaluations; we strongly recommend this approach to other researchers working with new interaction paradigms. Our software development life cycle could be considered a blend of user-centred design and extreme programming; frequent user evaluations keep a user-centred focus while continuous program development allows us to solve technical challenges.

Reliable recognition is essential if natural pen-only input is to be supported; anywhere, anyhow recognition is not reliable enough for our needs. The compromise we propose is software recognition limited to specific screen areas and input values, while the user must adapt their methods to work within these limitations.

FUTURE WORK

We have presented here a ‘work-in-progress’. We plan to support more input file formats so that essays and diagrams can be marked. We also have underway a more substantial usability study and a first study of the educational affects. We have received many suggestions from colleagues for additional functions; our experience with hand-drawn design software suggests that what is beneficial is often counter-intuitive. We see this software as another avenue to explore the benefits and limitations of hand-written interaction. This software also provides a platform to study interesting questions on effective educational communication strategies and computer mediated communication.

REFERENCES

Brush, A. B., Bargeron, D., Gupta, A., & Cadiz, J. (2001). Robust Annotation Positioning in Digital Documents, *Proceedings of Sigchi'01*, Seattle, WA, pp. 285-292.

Golovchinsky, G., & Denoue, L. (2002). Moving Markup: Repositioning Freeform Annotations, *Proceedings of Symposium on User Interface Software and Technology*, Paris, France, pp. 21-30.

Heinrich, E., & Lawn, A. (2004). Onscreen Marking Support for Formative Assessment, *Proceedings of Ed-Media*, pp. 1985-1992.

Heinrich, E., Wang, & Yuanzhi. (2003). Online Marking of Essay-Type Assignments, *Proceedings of Ed-Media*, Norfolk, USA, pp. 768 - 772.

Jarrett, R., & Su, P. (2003). *Building Tablet Pc Applications*. Redmond: Microsoft Press.

Marshall, C. (1997). Annotation: From Paper Books to the Digital Library, *Proceedings of DL*, Philadelphia, pp. 131-140.

Plimmer, B. E., & Apperley, M. (2003). Freeform: A Tool for Sketching Form Designs, *Proceedings of BHCI*, Bath, pp. 183-186.

Schilit, B. N., Golovchinsky, G., & Price, M. N. (1998). Beyond Paper: Supporting Active Reading with Free Form Digital Ink Annotations, *Proceedings of CHI 98*, Los Angeles, CA, pp. 249-256.

Shipman, F., Price, M., Marshall, C., & Golovchinsky, G. (2003). Identifying Useful Passages in Documents Based on Annotation Patterns, *Proceedings of ECDL*, Trondheim, Norway, pp. 101-112.

Wolfe, J. L. (2000). Effects of Annotations on Student Readers and Writers, *Proceedings of Digital Libraries*, San Antonio, TX, pp. 19-26.