Making Paperless Work

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

Mark Apperley
Department of Computer Science
University of Waikato
New Zealand
m.apperley@cs.waikato.ac.nz

ABSTRACT
Despite well documented advantages, attempts to go truly "paperless" seldom succeed. This is principally because computer-based paperless systems typically do not support all of the affordances of paper, nor the work process that have evolved with paper-based systems. We suggest that attention to users' work environments, activities and practices are critical to the success of paperless systems. This paper describes the development and effective utilization of a software tool for the paperless marking of student assignments which does not require users to compromise on established best practice. It includes a significant advance in the task management support.

Author Keywords
Paperless environments, affordances of paper, pen computing, annotation.

ACM Classification Keywords
H5.2. User Interfaces, *Input devices and strategies.*

INTRODUCTION
There are many good reasons why we might aspire to paperless environments, including reduction of the environmental harm of paper consumption and the economic cost of paper production, transfer and storage. Digital environments free us from the location and physical constraints of paper and provide better support for updating, filing, and comprehensive searching of documents. Yet, rather than decreasing, paper use is increasing in parallel with the increased use of computer systems.

Sellen and Harper [17, p76] suggest that there are four affordances of paper that are absent from most digital systems: flexible navigation; cross-referencing of multiple-documents; easy document annotation; and the modeless interleaving of reading and writing. However, they propose (p48) that for paperless environments to succeed, human systems must adapt to match the computer system. They evidence this with DanTech's relatively successful reduction in paper usage which they attribute to changes in work practices. We agree with Sellen and Harper that 'going paperless', as the two organizations they studied attempted, is doomed to failure in the short term. We believe this is because there is insufficient understanding of how paper supports various activities for that support to be replicated in computer software: additionally ideal computer hardware is not yet available. We argue that long term success will depend on computer systems better supporting human systems, and we demonstrate that this is possible.

There are two research challenges which must be met for such computer systems to emerge. First, we need to fully understand the role of paper documents in particular work practices [8]; this suggests an activity-centered model of analysis and design [9]. Second, technology must be developed (both hardware and software) to support equivalent paperless environments.

Clearly both of these are complex issues. Here we present a successful exemplar, Penmarked, a paperless environment for managing, annotating and grading student assignments. We suggest that our experiences will be of assistance to others developing paperless environments.

BACKGROUND
The use of paper is deeply integrated within our work practices; to convert tasks from paper-based to paperless we must understand papers essential qualities and uses. Some of paper's affordances need to be preserved in paperless equivalents, yet virtual paper can overcome the physical limitations of paper.

The materiality of paper documents is useful in a number of respects [8]. The physical dimensions and binding of a document convey meaning; a hard-cover book is very different in presentation from a hand-written note. The spatial position of documents on a desk may be significant, as may the order of the documents within a stack.

CHINZ 2007, July 2-4, 2007, Hamilton, New Zealand

Documents left on a work surface act as a reminder for an unfinished task. Document handover may signify a social process (eg the passing on of responsibility, completion, or collaboration) [6, 17].

However paper's predominant affordance for document storage is as an ink repository [17]; it represents an object from which we can read and upon which we can write, often interleaving these two activities. People prefer reading from paper to reading 'on screen'. A widely held belief is that this is because of the quality of screens and the fact that the projected light they emit makes reading more difficult than reading from paper. Yet, there are other characteristics of paper that are important. It is easier to quickly get an overview of the layout and content of a paper document by flipping through the pages [8]. Active reading includes activities such as scanning from the current point of focus backwards and forwards in the current document and between related documents. We often spread documents around our work surface and use spatial and tactile clues as markers to particular parts of the documents. For example we can arrange documents in a mind-map configuration or temporarily mark a passage by placing a pen on it.

Active reading includes annotating of the document or the writing of summary notes in a new document [17]; the act of annotation is a very powerful active reading and learning mechanism [21], and the resultant annotated document is a different artifact from the original document [18].

Although annotation techniques are both personal and idiosyncratic, annotations can be broadly categorized as emphases – achieved by highlighting, underlying, circling or side bars; corrections – crossing-out, overwriting, and inserting; or commenting – adding words or glyphs in the margins to explain or summarize the document [7].

Subsequent readers of the annotated document (including the annotator and the author) have a different reading experience [20]. They concurrently see the original work and the superimposed annotations. The annotations embellish the original document guiding the reader to points that the annotator considered significant.

There is a noteworthy difference between ink-over annotations on paper and the types of reviewing supported in word processors. With ink-over annotation one feels that the author still 'owns' the work as the original document remains fixed and the annotations are merely decorations. In contrast word processor reviewing alters both the appearance and layout of the original document so that authorship is now shared. Ink also affords powerful informal and iconic annotations that are difficult to replicate with keyboard and mouse [16].

On the other hand, the physicality of paper imposes limitations on paper documents that are not present in digital environments. First, the physical paper object requires physical creation, transportation, storage and destruction; each of these processes is significantly more expensive and time consuming than its digital equivalent [17]. Copying paper documents is also a physical process, while multiple copies of a digital document can be created, emailed and stored in digital form at a fraction of the cost. The fixed nature of the ink on paper documents makes substantial revision impossible, whereas editing a digital document is simple. Paper documents are, by their very nature, linear, while the natural structure of some documents we may wish to represent is non-linear (for example object-oriented programs). A digital environment better supports non-linear navigation, and the indexing and searching of documents and document repositories.

However, electronic environments do impose different constraints, such as the requirement for suitable hardware, reliable power sources and network connectivity. Much of this technology is currently available in the office, but is not yet as mobile and 'always on' as paper. Marshall [8] attributes the failure of e-books to the need for specialized devices, and suggests that to be successful, this technology must be supported by appropriate software on general computers.

In fact, although we think of paper documents being a more permanent record of a document than their digital equivalent, a single copy of a paper document is more expensive and far more fragile than the multiple copies that usually exist of digital documents. Consider for example the various versions and revisions of this paper; the cat chewed hand-written notes left on the floor, but there are probably 4-6 digital copies of it, on our hard drives, mail servers and backups.

Assignment marking

Our goal with this research is to explore the requirements for the complete and effective replacement of a paper-based document system. We do this through the development of an exemplar, a computer-based environment for marking student assignments. Traditionally assignment marking is a paper-based activity. The student completes his or her assignment and submits the script to the teacher. The *marker* (teacher or teaching assistant (TA)), reads, annotates, ranks and grades the papers, records the grade in a grade-book, and returns the script to the student. Thus, paper is employed for a range of common reading, writing and document management tasks.

Below we provide a detailed description of assignment marking. We contend that an in-depth understanding of the paper-based activity is an essential precursor to the design of a paperless environment. Our approach is a combination of autoethnography [1] and a review of earlier work.

Traditionally students handed their paper 'scripts' to the teacher. In many cases this has already been replaced by electronic submission [for example, 5, 13, 14]. Electronic submission is easier for the student, and alleviates problems of misplaced scripts and the time-stamping of submission. Further, when the assignment is a computer program or similar, a digital copy of the work is required for grading purposes.

The teacher may quickly scan the entire collection of scripts before commencing grading. A survey of colleagues suggested a number of preordering techniques they employ. Sometimes they will select the scripts that they expect to be best and/or worst and inspect, but not necessarily grade, these first. On other occasions they may arrange the scripts in a ranked order before assigning grades, or if the assignment topics vary then the scripts may be grouped by topic before grading commences. Both scanning and sorting document collections are facilitated by paper scripts, but can be difficult in standard digital environments.

Our observations of teaching assistants (TAs) suggest that they take a more linear approach. In our departments the teacher provides a detailed marking rubric for the TAs, who, we have observed, start at the top of the stack and mark each assignment in turn against that rubric. They rarely compare scripts or undertake any scanning or pre-ordering, overview activities in which they have little interest.

The next and most important step in the marking process is a careful examination of each student's work. Traditionally the marker annotated the work with a *red pen*. Such annotations serve three purposes. First, they act as a reminder to the marker of the critical points in the assignment; it is common practice to use annotations as a part of the grade decision process. Second, the ink records the summative grade assigned to the work. Finally, and most importantly, annotations provide feedback to the student.

Individualized formative feedback on a student's work is acknowledged as a powerful learning intervention [14]. All manner of electronic grading systems have been proposed that can produce a fair summative grade [for example, 5]. The most sophisticated may generate specific feedback depending on the errors detected; however, the personalization is lost. The ability to easily annotate a script is a capability that is needed in order to achieve a successful paperless system in this application.

To move from one script to the next when working with paper is simply a matter of lifting the next script off the stack. Many paperless environments require significantly more effort, including the closing and opening of folders, applications and documents.

Once a grade has been determined for a script, the next step in the marking process is the recording of the student's grade both on the script and in a grade-book. The modern trend is to grade against a set of criteria and allocate a score to each criterion. The grade is then derived by calculating a weighted average of the criteria. A paper-based system requires manual calculation of the grade and its double entry on the script and in the grade-book. In contrast, with a computer-based system, both calculating the grade and producing multiple copies of it is trivial.

The final step is return of the script to the student. The options for return of paper assignments are governed by the physicality of paper; for example the assignments can be distributed in class, held somewhere for collection or (snail) mailed to the students. In contrast, electronic systems can accomplish this automatically, either by email or placement of the graded assignment into a digital repository. Furthermore, with paper, there is only one copy of the assignment; if it is lost, it is lost forever, or if the teacher is required to retain a copy (for example, for moderation) then it must be photocopied.

From this description of marking as an activity it can be seen that the challenges in creating an electronic paperless system for this application are support for work tasks such as moving effortlessly between scripts, freehand annotation of scripts, and simultaneous recording of scores. Our objective is to provide both faculty and students with 'the best of both worlds'. This is to say, an electronic environment that provides rapid, easy storage, transmission of documents and recording of grades, and at the same time an informal paper-like environment that affords quick commenting and rich feedback.

DESIGN STRATEGIES

As computer scientists, most of the assignments our students write are computer programs. For us, detailed and in situ comments are as important in marking programs as in the case of more traditional essay-style assignments.

Our initial focus was on supporting the marking of programming assignments; this focus has both pros and cons. The advantages are that our users (programming teachers and TAs) are expert computer users accustomed to, and unfazed by, new technology. Also as programming is predominately a computer-based activity, the markers expect to use a computer for at least part of the grading process. The disadvantages are three-fold; program code is often saved in multiple files (one per program class); programs usually need to be compiled and executed as a part of the marking process; and programs may have a complex non-linear structure, in comparison with essays, which are typically read in a linear fashion. However, we also intend this prototype to be useful for grading more traditional assignments such as essays.

Norman's [9] activity-centered model of an activity, task, action, hierarchy is used as a framework for this design discussion. The activity that we are concerned with is the marking of a set of student assignments, with special consideration for marking programming assignments. The set of tasks involved in this activity are collecting assignments, pre-marking preparation, detailed examination and annotation of each assignment, allocation of scores, archiving of grades for the grade-book, and return of the scripts to the students. There are tools and techniques readily available for each of these steps – however many markers rely on paper for part of the process because the connections between the parts is not well supported or the formal nature of the computer tools limits information transfer. More detailed technical descriptions of Penmarked

are available [10, 11], here we concentrate on the support for paperless work.

Most institutions have student support systems that include electronic drop-box facilities. We propose a flexible assignment collection architecture that will support either a web-based service or a customizable interface to an electronic drop-box. Data such as student identifiers, contact information and assignment file information, which is required for marking, is customarily available from such facilities. Essay assignments are usually submitted as a single file, while computer programs are frequently submitted as a zipped package of multiple files. In this latter case, only selected files may be of interest to the marker. A general marking system must deal with both these situations, ie single and multiple file scripts.

Pre-marking activities include the preparation of a marking rubric and pre-scanning and ordering of scripts. A wide range of marking rubrics may be employed, from a simple alphabetic grade (A+) to complex multi-criteria Likert scales. These are easily accommodated in an electronic system, although, as our intention is to support pen-based interaction for annotation, recognition of handwritten scores is desirable.

Providing an interface that makes it as easy to pre-scan multiple scripts as it is to scan paper copies is a challenge with current technology [8]. A large wall or tabletop display could show multiple scripts on the screen in such a manner that the marker could scan and sort them; interaction techniques for flicking through digital documents have been explored for use with digital books [16]. However, to realize this design as a prototype, we restricted ourselves to working with standard hardware such as a desk-top, lap-top or Tablet PC. A viable software approach with such environments would be to list all the scripts, and to allow quick inspection and ordering by clicking and dragging actions on this list.

The detailed examination of each script relates directly to Sellen and Harper's [17] notion of active reading. The reader needs to be able both to read and annotate simultaneously; written feedback is a vital ingredient of assessment. Some have proposed off-line comments, however these are clumsy and time-consuming to construct because a point of reference must be established (e.g. on page 3, paragraph 2...). Heinrich and Lawn [3] propose typed annotations that lie over the document. While this places the comments in situ, typed comments lack the flexibility and expressiveness of ink. We suggest direct inking on to the document on a Tablet PC monitor [16].

Grading computer programs calls for particular software functionality [14], including provision for multiple file scripts, and navigating between these, as well as provision for program compilation and execution. The marker needs access to the program code in both the marking software (which supports annotation and the recording of scores) and the programming environment (where they can execute the program); our solution to this is the use of two monitors: a tablet for annotating the script and a standard monitor to display the programming environment (Figure 1).

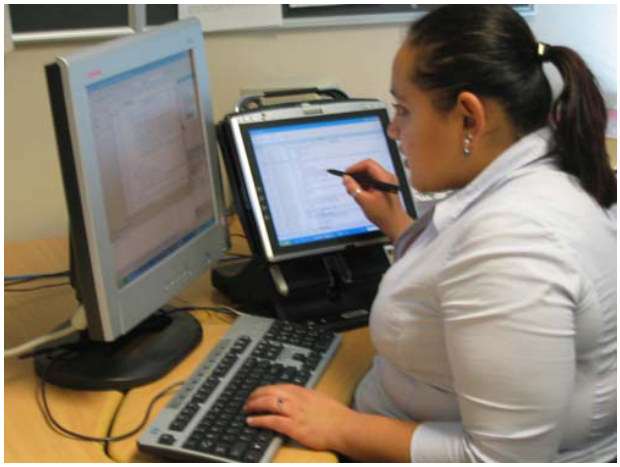

Figure 1. Penmarked in use with dual monitor setup.

Rating the assignment against criteria is integrated with active reading and feedback annotation. Scores must be assigned to the work and recorded for use in a grade-book. The ideal is for the marker to write scores once only. Current recognition techniques could not reliably disambiguate a score a written anywhere on the document from other annotations. We considered three possibilities to achieve score recognition. First, the marker changes ink modes to write the score – this was discarded because modal inking has been shown to be less than satisfactory for users [12]. Second, a score zone (a square in one corner or column down one side) is provided on the document and any annotations in this zone are recognized as scores. Last, a separate table is provided which contains a marking schedule, into which markers can record scores with either a stylus or keyboard. However inking accurately requires a larger space than a type-written score takes, so to keep the space for a score table small we suggest a separate inking zone adjacent to the table. The marker writes a score in the inking zone which the system recognizes and stores in the table. Once recognized, scores can be totaled and archived.

The final marking tasks are the saving of the grades into a grade-book and returning the scripts to the students. As with collection of the scripts, most institutions have existing student databases; saving grade-book information in standard file format supports integration with other systems. For reliability purposes the scripts need to be saved in a fixed format that makes them difficult to change (this may be contrary to the requirements of many other document annotation activities). Two alternatives for return of work are emailing and a reverse drop-box, depending on standards and protocols for the particular institution.

PENMARKED PROTOTYPE

Figure 2. Penmarked main interface in default layout.

Penmarked (Figure 2) has been developed for use on a tablet PC running the Microsoft XP Tablet™ operating system and the Visual Studio .Net Framework™. To meet the design goals described above, the major focus of the prototype development has been ensuring that all stages of the assignment marking activity are appropriately supported. The key components of the system are: management of pre- and post-marking activities for a class set of assignments; on screen annotation of scripts and concurrent recording of scores; and, for marking computer programs, easy transition between Penmarked and a programming environment.

Pre-Marking Activities

Penmarked support starts with the setting up of an assessment, with a wizard to step the user through this process. First the user defines the marking schedule (rubric); they may load an existing schedule and modify it, or create an entirely new schedule. The schedule specifies the assessment criteria and minimum and maximum score for each criterion.

Next the user specifies the location of the student files. The data that the system requires are a student identifier and email address (for return of work), and the name(s) and the location of submitted files. As the format of electronic drop-boxes varies widely we have implemented two plug-ins. One is tailored to a locally developed drop-box and another based on XML files. In either case, the specified directory structure is parsed and information gathered to add each assignment to the student list for display.

The student list has become progressively more complex with each prototype iteration. Initially it was a simple list in alphabetic order. We then added check boxes so the marker could check-off assignments as they are

completed. Later we added a change of colour and font for assignments that had been opened but not completed. More recently teachers (rather than TAs) have used Penmarked. In one case the students had a choice of essay topic; the teacher would have liked to be able to preorder the assignments by topic. We foresee no difficulty implementing this functionality; we report it here to emphasize the importance of replicating all of the affordances of a stack of papers.

Marking an assignment

To mark an assignment the marker selects the student's list entry. Penmarked locates the student's submitted files; zip files are extracted and placed into a subdirectory and the location of the student's files is associated with the folder icon on Penmarked's toolbar. A user defined filter is used to select appropriate files (e.g. all those with a particular file extension except *xyz*). We found that file type and name filtering is particularly important for marking .Net programs as the environment generates a large number of files in which the marker has little interest, but which are required to compile and execute the program. This is an example of how closely the paperless environment must be tailored to the task.

Selected text and rich-text files are displayed in the annotation pane (Figure 3). If there is more than one file, each is displayed in a separate tab. The marker can now read, annotate and score the script against the marking rubric. To navigate through the script the marker can scroll within a window using the scroll bars, or jump between windows using tabs. A find dialogue proved to be useful for locating specific points in the programs.

Ink annotations can be placed anywhere on the documents or in an auxiliary panel. The ink is held on a transparent overlay that is superimposed over each document. Positioning between the two layers is synchronized so that they appear as one. An ink eraser is also provided; users have commented on this as a real benefit of a digital environment. The underlying script cannot be changed by the marker.

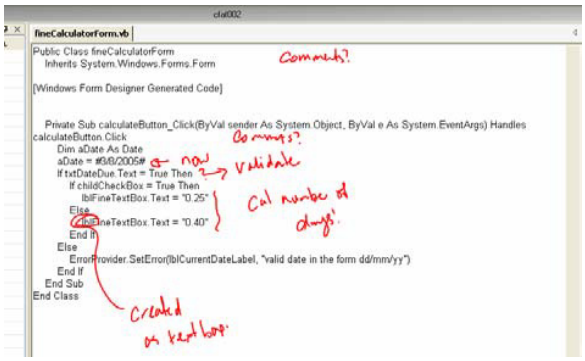

Figure 3. Assignment annotation pane

The mark schedule (Figure 4) is created for an assignment when it is first opened in Penmarked. The marker navigates the marking rubric using the scroll bar and can select a criterion to be scored. The score is entered by either writing in the input box or via the keyboard. Ink input is recognized using the operating system recognizer. The input is validated against the minimum and maximum, and if valid is stored in the mark table. If the input is invalid the user is alerted to this fact.

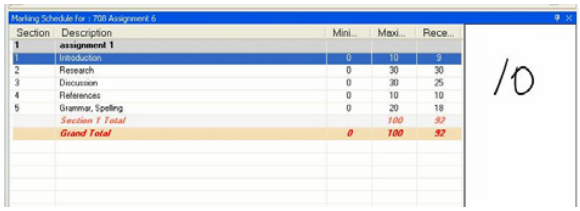

Figure 4. Marking schedule showing a score in the input box

When marking programs the source files are accessed by selecting the folder icon on the Penmarked toolbar. This opens the Windows folder that contains the student's file. From here the marker can open the programming environment and compile and execute the program. We noticed that markers switch between the programming environment and Penmarked frequently, as is typical for a multi-document exercise [17]. We have implemented multiple display support in Penmarked, with the tablet screen used to display the marking software so that the marker can annotate with a pen, while the second monitor is used to display the folder of student work and the programming environment (Figure 1).

We carefully designed Penmarked to fully support pen-only interaction. However, when marking programs that are not designed for the tablet, usability testing showed that it was much quicker to interact with student's (non tablet) programs with a keyboard.

Post-marking tasks

When marking is finished the marker selects the assignments for return, and chooses the method of return. The system generates a PDF file for each student that contains both the completed marking rubric and the annotated assignment documents. The PDF file is emailed to the student via either STMP or Microsoft Outlook™. The method of return and mail client information is specified in the Penmarked options. Grades and identification data are exported in XML format to provide maximum flexibility for interfacing with other software.

Interaction design

Penmarked has been designed with pen-only interaction in mind [4]. Access to the frequently used functions is via the main interface toolbar. The interface is

configurable with both the student list and marking schedule held in floating, resizable and dockable windows. These windows can be stacked on top of each other; in this case tabs appear for the user to navigate between the two. This provides maximum flexibility for users to adjust the viewing spaces and adaptation for right or left-handedness and screen orientation.

Stylus entry of scores requires clear writing for successful recognition; this is easier to achieve when writing with larger gestures. In situ writing in the marking schedule was not used because of the screen space that would be required. The XP on-screen keyboard was discarded because it would reduce the screen space available to Penmarked or require frequent opening and closing. The panel we have implemented is a balance between screen-space, accurate recognition and unnecessary interruptions of the user. Potentially users could allocate marks to the wrong category or not notice when a mark was misrecognized. We analyzed usability testing recordings of 12 assignments being marked by 4 different markers, carefully observing this task. There were no incidents of markers allocating a score to the wrong criterion. There was one incident of a marker not seeing that a score was misrecognized. The system has been modified to check that each marking schedule is completed before allowing the return of the work to the student. This is still fallible; we are exploring other options to fail-safe this feature without interrupting the user with 'ok cancel' dialogues.

Penmarked supports a range of hardware configurations. It can be used in tablet only mode, where all interaction is via the stylus. This works extremely well when marking essays and provides maximum portability. It can also be used on a tablet with a keyboard attached, as users have found a keyboard is useful for data entry when marking programs. Finally, it can be operated with a tablet, keyboard and second monitor, providing the optimum configuration for marking programming assignments, as Penmarked and the programming environment can be seen and interacted with simultaneously.

Penmarked has been extensively evaluated in use. Early in the development we undertook numerous ad hoc small evaluations with colleagues and TAs. We ran three formal usability tests, a think-a-loud study and focus group [self reference] when we had completed our first prototype, and a lab observation using Morae™ [19] on our second prototype. Penmarked has been used for marking about one thousand student scripts, over six different assessments. The types of assignments include Visual Studio programs, Java programs and essays. Twelve different markers (both teachers and TAs) have been involved in these trials. They have reported that they do not print anything or feel any need to print the documents.

DISCUSSION

The Penmarked prototype addresses many of the affordances of paper that are not available in standard systems. First, Penmarked makes it very easy to navigate from script to script, and between script and programming IDE. It is also easy to move between multiple documents that make up a single assignment script. The 'find' functionality implemented in Penmarked is an example of search that is not available on paper.

The marker switching between Penmarked and the programming IDE is an example of multiple document referencing [17]. The two-monitor environment provides a satisfactory solution to this problem. An alternative possibility would be to integrate the functionality of Penmarked into a programming IDE.

The interleaving of reading, writing and erasing are fully supported. There are two distinct forms of writing, annotation onto the script, and the entering of scores. We have observed markers interleaving reading, both forms of writing, and program execution without difficultly.

Many potential users have asked whether there is a clipboard for comments. This contrasts with the observations of [7] that annotations are unique and idiosyncratic. Also, a pasted comment would require manual positioning and resizing [2]. This could take longer than creating a new annotation, but it an area worthy of investigation.

Marking student scripts involves not only reading, commenting on and grading but also pre- and post-marking tasks. Through sustained use of this system, and discussions with colleagues, we have learnt much about the types of tasks that people undertake both before and after marking a set of scripts. The student list has undergone a number of reviews during our prototype development and we can see potential for improved functionality, further enhancing the paperless system by some of the affordances offered by computerization; for example, the ability to reorder the list as we commented above; adding comments related to a student; or showing the total grade once the assignment has been marked.

We noted that there are differences between how TAs and teachers mark assignments. Similar differences are likely to exist with other activities depending on the user's role and individual preferences.

One task associated with marking computer programs is not well supported in Penmarked, compiling and running the program. As a foray into this we implemented ink annotation into the Visual Studio IDE [15] and while this shows promise as an idea, the environment proved to be very difficult to program. We are currently exploring other environments.

CONCLUSIONS

Penmarked supports educational best practice by providing in situ ink annotation of a student's work and a marking rubric against which the marker can score the work. Through formal evaluation studies and on-going use of Penmarked, our understanding of the affordances of paper that are important to this activity have been honed.

Many attempts to move to paperless environments have been unsuccessful. Often this is because the replacement computer systems do not support the appropriate paper affordances required for the activity. With this research we began by examining the differences between paper-based and standard computer-based document activities and then focused specifically on assignment marking as an activity. We have described this in detail, as we assert that it is only with a comprehensive understanding of the tasks involved in a paper-based activity that a paperless system can be designed to support it. It is necessary to identify the paper affordances that are critical to the task to those that can be better replaced with digital equivalents.

Assignment marking is just one example of a traditionally paper-based activity that can now become paperless. The technology is available; by unraveling the entanglement of paper in our work practices we can understand which affordances of paper to retain and which to replace. Penmarked, like all systems, is constrained to technology of today. As larger stylus sensitive surfaces become available the desk-top metaphor may become a reality. We can imagine a digital desktop where we can indeed stack documents and spread them around, annotate and edit them, in a totally paperless, yet natural work environment.

REFERENCES

1. Duncan, M., Autoethnography: Critical appreciation of an emerging art, International Journal of Qualitative Methods, 3, 4, Article 3, (2004),
2. Golovchinsky, G., Denoue, L., Moving markup: repositioning freeform annotations, in proc UIST '02, ACM, (2002), 21-30
3. Heinrich, E., Lawn, A., Onscreen marking support for formative assessment, in proc Ed-Media, (2004), 1985-1992
4. Jarrett, R., Su, P., Building Tablet PC applications, Microsoft Press, (2003),
5. Joy, M., Luck, M., Effective Electronic Marking for On-Line Assessment, in proc ITiCSE, ACM, (1998), 134-138
6. Liu, Z., Stork, D. G., Is paperless really more?, Communications of the ACM, 43, 11, (2000), 94-97
7. Marshall, C., Annotation: from paper books to the digital library, in proc DL, ACM, (1997), 131-140
8. Marshall, C., Reading and Interactivity in the Digital Library: Creating an experience that transcends paper, in proc CLIR/Kanazawa Institute of Technology Roundtable, (2003), 5.4.1-20
9. Norman, D., Human-Centered design considered harmful, Interactions, (2005), 14-19
10. Plimmer, B., Mason, P., Designing an Environment for Annotating and Grading Student Assignments, in proc OZCHI, (2004), 45-53
11. Plimmer, B., Mason, P., A Pen-based Paperless Environment for Annotating and Marking Student Assignments, in proc AUIC, CRPIT, (2006), 27-34
12. Plimmer, B. E., Apperley, M., Software for Students to Sketch Interface Designs, in proc Interact, (2003), 73-80
13. Preston, J. A., Shackelford, R., Improving on-line assessment: an investigation of existing marking methodologies, in proc ITiCSE, ACM, (1999), 29-32
14. Price, B., Petre, M., Teaching programming through paperless assignments: an empirical evaluation of instructor feedback, in proc ITiCSE, ACM, (1997), 94-99
15. Priest, R., Plimmer, B., RCA: Experiences with an IDE Annotation Tool, in proc CHINZ, ACM, (2006), 53-61
16. Schilit, B. N., Golovchinsky, G., Price, M. N., Beyond Paper: Supporting active reading with free form digital ink annotations, in proc CHI 98, ACM, (1998), 249-256
17. Sellen, A. J., Harper, R. H. R., The myth of the paperless office, MIT, (2002),
18. Shipman, F., Price, M., Marshall, C., Golovchinsky, G., Identifying useful passages in documents based on annotation patterns, in proc ECDL, (2003), 101-112
19. TechSmith, Morae, 2004, (2005),
20. Wolfe, J. L., Effects of annotations on student readers and writers, in proc Digital Libraries, ACM, (2000), 19-26
21. Yang, S. J. H., Chen, I. Y. L., Shao, N. W. Y., Ontology enabled annotation and knowledge management for collaborative learning in a virtual learning community, Educational Technology & Society, 7, 4, (2004), 71-81