GEORGE FOX UNIVERSITY

Digital Commons @ George Fox University

Faculty Publications - Department of Electrical Engineering and Computer Science

Department of Electrical Engineering and Computer Science

2013

Paperless Subjective Programming Assignment Assessment: A First Step

David Hansen
George Fox University, dhansen@georgefox.edu

Follow this and additional works at: http://digitalcommons.georgefox.edu/eecs_fac


Part of the Computer Sciences Commons, Educational Methods Commons, and the Higher Education Commons

Recommended Citation

Hansen, David, "Paperless Subjective Programming Assignment Assessment: A First Step" (2013). *Faculty Publications - Department of Electrical Engineering and Computer Science.* Paper 4.
http://digitalcommons.georgefox.edu/eecs_fac/4

This Article is brought to you for free and open access by the Department of Electrical Engineering and Computer Science at Digital Commons @ George Fox University. It has been accepted for inclusion in Faculty Publications - Department of Electrical Engineering and Computer Science by an authorized administrator of Digital Commons @ George Fox University. For more information, please contact arolfe@georgefox.edu.

PAPERLESS SUBJECTIVE PROGRAMMING ASSIGNMENT ASSESSMENT: A FIRST STEP

David M. Hansen
Computer Science and Information Systems Department
George Fox University
Newberg, OR 97132
503 554-2709
dhansen@georgefox.edu

ABSTRACT

A significant component of student evaluation includes the *objective* and *subjective* assessment of programming assignments. We describe bits and pieces of a paperless electronic workflow we've recently used to provide objective and subjective feedback to students, emphasizing the tools and process used to provide *subjective* programming assignment assessment. We identify opportunities for future work to complete an end-to-end paperless electronic workflow that incorporates subjective assessment.

MOTIVATION

Don Knuth famously stated "The chief goal of my work as educator and author is to help people learn how to write *beautiful* programs" [3]. We share Professor Knuth's passion, believing that *how* a programming assignment is solved by a student is as, if not more important than that it is solved. Thus the *subjective* assessment of student programming assignments is a vital component in teaching them how to write beautiful programs.

Historically, student programming assignments, along with evidence of correct functionality, were submitted for assessment and grading via hardcopies that were marked-up by faculty and returned to students. As technology progressed, email submission of assignments became common. More recently, electronic submission of assignments in general has become commonplace throughout the University as many "course management systems" provide mechanisms for web-based assignment submission. A number of faculty at our University have gone "paperless" and use tools such as the Apple iPad to assess, grade, and return assignments submitted electronically. However, programming assignments must generally pass a test that your average "essay" does not; programs must perform correctly. A number of frameworks, or even extensions to existing "course management systems", have been implemented to accept programming assignments and automatically conduct a variety of objective tests and assessments [1, 2, 5]. However, *subjective* assessment generally is still accomplished by printing programs and returning marked-up hardcopies to students.

AN IDEAL WORKFLOW

Consider an entirely electronic and paperless system where:

1. students could submit their assignments, both non-programming and programming (receiving immediate feedback if the programming submission could not be tested in the following step),

2. the programs would undergo *objective* testing and analysis,
3. **both non-programming and programming assignments, together with the output of the previous step, could be annotated with *subjective* feedback,**
4. grades recorded for the assignments, and
5. the entire assignment, together with objective and subjective feedback, be returned to the student.

Although it is one of the most vital steps in assessing student work, step 3 in this workflow remains the weakest link in existing frameworks. One notable system, Web-CAT, essentially provides a sort of end-to-end online system with automated objective assessment as well as a web-based interface for providing subjective assessment of program submissions [4]. It may well be that Web-CAT is a viable solution. However, its stated purpose is to "grade students on how well they test their own code" [4] which is not quite the purpose of what we've outlined above. Furthermore, Web-CAT's web-based interface for subjective assessment doesn't meet the desired goal of electronic "pen and paper" and does not permit working offline.

Before investing significant time in developing and fully automating such a workflow or deploying a tool such as Web-CAT, we wanted to investigate and test hardware and software that might be used to accomplish step 3. We selected an entry-level Apple iPad and the PDF markup application *iAnnotate* to test the ease and efficacy of conducting step 3 in a paperless fashion.

A FIRST STEP

We chose to attempt as seamless a migration from hardcopy paper-based assessment to paperless electronic assessment as possible. Our current "workflow" accepts assignment submissions via email – both programs as well as non-programming "book work" assignments. Programs are generally dumped into a directory, one-by-one; a shell-script runs a series of automated tests, collects the output, and prints the source-code together with the output to hardcopy; this is combined with hardcopy of any "book work" component of the assignment. The following is a typical script that tests a student's linked-list implementation using a JUnit test harness and prints the program source together with output of the test harness:

```
javac test.java
javac -cp ./ MyLinkedList.java >& compile.out
java test >& test.out
enscript -B -C -E -f Courier7 MyLinkedList.java
enscript -B compile.out
enscript -B -f Courier8 test.out
rm –rf *.class *.out MyLinkedList.java
```

Subjective assessment and feedback have been given via pen and paper, with the hardcopy returned to students during a subsequent class session. As students value, if not always appreciate, the subjective feedback provided on programming assignments, the natural choice was to use hardware and software that allowed us to continue to provide feedback similar to "pen and paper" mode where "pen" and "paper" were electronic.

We began with a minor modification to our workflow. At the point where programs were printed we instead "printed" to a PDF file. Since the platform where programs are objectively tested (a desktop or laptop) is not the same as the platform being used for assessment (iPad), we leveraged Google's GoogleDrive system to provide a convenient shared location for the PDF files. The previous shell-script became:

```
javac test.java
javac -cp ./ MyLinkedList.java >& compile.out
java test >& test.out
enscript -C -E --color=1 MyLinkedList.java -o - | ps2pdf - .c.pdf
cat compile.out test.out | paps --paper letter | ps2pdf - .o.pdf
pdftk .c.pdf .o.pdf *.pdf cat output ~/GoogleDrive/`date +%s`.pdf
rm –rf *.class *.out *.pdf .c.pdf .o.pdf MyLinkedList.java
```

Additionally, students were asked to include "book work" assignments as PDF attachments to their email that can be dropped into the same directory and "printed" together with the program and output into a single PDF file (thus the `*.pdf` argument to `pdftk` in the script above). One additional benefit of grading program source code printed to PDF files is that utilities such as `enscript` can generate language-sensitive, color-coded output of program source; the example script above generates Java-specific output that italicizes comments and uses color to distinguish language elements such as keywords where non-color hardcopies could not. Once all programs have been tested and PDF files generated, we use *iAnnotate* on an iPad to grade and subjectively assess programs and other assignments. After all grading is complete, the annotated PDF is emailed back to the student.

Although workable, the current workflow could be simplified and improved if the connection between the initial submission, the annotated PDF, the student in our gradebook, and the student's email address were maintained throughout the process. For now, these are discrete steps manually taken – the very thing a "course management system" excels at automating.

INITIAL ASSESSMENT

We were initially skeptical about moving to a paperless workflow for grading programming assignments. In our experience, hardware and software tools often fall short of their promise and complicate, rather than simplify or enhance a process. This explains why we began this journey by making minimal changes to our existing manual workflow. We intended to get a hands-on sense of the maturity of the tools we would be using in step 3 above before proceeding further; if paperless grading proved too tedious, no time or effort would have been wasted on implementing or configuring an end-to-end workflow.

We have now used the paperless workflow described above in four different upper-division Computer Science courses. The courses all include non-trivial programming assignments and one course included a substantial formal written assignment. We can say without reservation that

- we have no plans to return to paper-based grading and
- we are now eager to adopt, adapt, and or develop tools to complete the sort of fully paperless and seamless electronic workflow described earlier.

Benefits

There are a number of benefits to paperless grading – some, unanticipated. Practically speaking, within two years we will have saved enough money to pay for the iPad and associated software by not printing each and every assignment (a representative course last semester had the equivalent of 1465 printed pages of programs and homework submitted) – color us green!

More significantly, tools such as *iAnnotate* provide multi-modal feedback. It was this opportunity to move beyond handwritten notes that initially spurred us to consider moving to an electronic platform – it would not simply be a *different* means of providing feedback, but a *better* means. PDF annotating tools allow us to continue to provide "handwritten" feedback, but we can also add typewritten notes and include audio annotation. Handwritten comments can be constraining – it is hard to provide much feedback within the margins of a page. Typewritten notes are useful for programming assignments where we may want to provide an alternative coding example that may be too detailed to write by hand. Even better, when we have lengthy feedback to provide, we can attach an audio annotation that is more efficient to dictate than to write. It is a frequent lament that students do not avail themselves of office hours often enough, so an audio annotation gives us the opportunity to express verbally what we might tell a student about an assignment if they were to visit our office.

An unanticipated benefit of paperless grading is that assignments can be returned to students at any time instead of waiting until the next class meeting. Though generally prompt about returning graded assignments, the lag between submission and feedback can be as much as 5 days for a class meeting, say, Tuesday and Thursday. And if a subsequent assignment depends on the previous assignment, the delay in feedback can have unfortunate consequences. Under the paperless workflow, graded assignments can be returned to students via email immediately, sometimes days before the next class session.

Another unexpected benefit is the "undo" nature of electronic grading. We will occasionally reconsider written feedback, scrawling it out on hardcopies and, in extreme cases, reprinting pages to "erase" an inaccurate assessment entirely. Electronic grading eliminates this problem, allowing one to delete or amend comments up until the moment the graded assignment is returned to the student.

Practical Considerations

We were initially concerned that we would find it difficult to move from hardcopy paper to an electronic platform. We confess to being dinosaurs, preferring tangible books to their electronic substitutes. However, we found the transition to electronic "pen and paper" relatively easy. After getting used to the platform and tools, it took no longer to grade and annotate work electronically than it did via paper. Paperless grading is not without limitations, however.

We belatedly came to appreciate the difference between the "resistive" touchscreen of our mobile-phones and the "capacitive" touchscreen of the iPad. The capacitive screen of the iPad is optimized for human fingers and a traditional pen-like stylus simply does not work. The iPad specifically requires a stylus designed for a capacitive touchscreen - styli that are blunt-tipped, unlike the pen-like stylus we had expected to use. Furthermore, our initial styli were inexpensive devices procured at a local grocery store and, over time, demonstrated that "you get what you pay for," becoming balky and unresponsive. We have since obtained two other styli that are highly recommended for the sort of writing tasks that characterize this application [6, 7] and they dramatically improve the experience. Still, the mechanics of "writing" on a capacitive touchscreen take some getting used to.

Additionally, the PDF standard is somewhat vague and inconsistent. Some of the

markup mechanisms available via *iAnnotate*, for example, are not visible when the PDF file is viewed with the default OS/X PDF viewer *Preview*. Fortunately, Adobe's free and ubiquitous *AcrobatReader* is compatible with all of the mechanisms provided by *iAnnotate*.

Student Assessment

Students in the four courses for which we used paperless grading were invited to provide feedback on the process via a brief informal poll. The poll asked the following questions comparing paperless to previous paper-based grading:

1. Amount of feedback received via paperless grading
2. Quality of the feedback via paperless grading
3. Turnaround time of assignments via paperless grading
4. Audio annotation usefulness
5. Overall assessment of paperless grading

Interestingly, the amount of feedback received tended to be slightly *less* via paperless grading (2.9 on a 5-point scale ranging from "much less/worse", "less/worse", "same", "more/better", "much more/better") yet the quality of the feedback was scored *higher* (3.4). The most significant change was in turnaround-time which was ranked quite a bit better (3.8 with no score lower than "same"). Of the 40% of respondents who received an audio annotation on an assignment, the usefulness was rated high (3.9), affirming our sense that audio annotation is a powerful mechanism that enabled better feedback, encouraging us to make audio annotations more often in the future.

Overall, students seemed pleased with the move to paperless grading, giving an overall average score of 4.15; 85% rating paperless grading a 4 or 5 with no ratings below 3.

Students were also invited to provide their own list of benefits and drawbacks. Many student-observed benefits were consistent with our own impressions: faster turnaround, audio annotations that were more detailed than written, and no wasted paper. One benefit we had not considered was that many students were much happier with the portability of the electronic format; no binder of assignments to maintain and they could instantly access and refer back to their previous work. Portability was an unanticipated benefit for faculty as well. We no longer needed to have a backpack loaded with paper assignments with us when the urge to do some grading struck, allowing us to engage in incremental grading at times and in places that were traditionally inconvenient.

The only drawbacks students cited were an occasional issue with the legibility of handwritten notes (not uncommon on hardcopies as well) and the requirement to use Adobe's *AcrobatReader*.

CONCLUSIONS

Most of the bits and pieces of the idealized electronic workflow described earlier already exist and merely await some "plumbing" or "customization" for bringing both *objective* and *subjective* assessment of student programming assignments together into a single, seamless and integrated workflow. But before pouring significant effort into assembling and deploying a framework, we wanted to assess the viability of an important part of the process – paperless subjective grading.

Through this process we have become converts to paperless grading. In the near-term we will continue to incrementally adapt our workflow to take email submissions, use automated testing for *objective* assessment where applicable, and generate PDF files for *subjective* assessment and grading. Going forward we plan to reconsider deploying a tool such as Web-CAT, especially if it can be customized to allow the sort of offline electronic "pen and paper" assessment we have employed using the iPad, or looking to build out our own end-to-end paperless grading workflow.

REFERENCES

[1] Cheng, Z., Monahan, R., Mooney, A., nExaminer: A Semi-automated Computer Programming Assignment Assessment Framework for Moodle, International Conference on Engaging Pedagogies, December 16, 2011.

[2] Hayes, A., Thomas, P., Smith, N., Waugh, K., An investigation into the the automated assessment of the design-code interface, 12th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2007), June 25-27, 2007.

[3] Knuth, D., Literate Programming, Lecture Notes (27), Stanford, CA: Center for Study of Language and Information – CSLI, 1992.

[4] Popyack, J.L, Herrmann, N., Char B., Zoski, P., Cera C., Lass R., Pen-Based Electronic Grading of Online Student Submissions, Drexel University, Presented at the Syllabus fall2002 Boston Area Conference on Education Technology, Newton, Massachusetts, November 4-5, 2002.

[5] Shamsi, A., Elnagar, A., An Intelligent Assessment Tool for Students' Java Submissions in Introductory Programming Courses, Scientific Research Publishing, 2012.

[6] http://www.wacom.com/products/stylus/bamboo-stylus

[7] http://the-maglus.myshopify.com/

© CCSC, (2013). This is the author's version of the work. It is posted here by permission of CCSC for your personal use. Not for redistribution. The definitive version was published in The Journal of Computing Sciences in Colleges, 29, 1, October 2013, http://dl.acm.org/.