Computerized Testing: A Vision and Initial Experiences

Prof. Craig Zilles, University of Illinois at Urbana-Champaign

Craig Zilles is an Associate Professor in the Computer Science department at the University of Illinois at Urbana-Champaign. His current research focuses on computer science education and computer architecture. His research has been recognized by two best paper awards from ASPLOS (2010 and 2013) and by selection for inclusion in the IEEE Micro Top Picks from the 2007 Computer Architecture Conferences. He received the IEEE Education Society's Mac Van Valkenburg Early Career Teaching Award in 2010, a (campus-wise) Illinois Student Senate Teaching Excellence award in 2013, the NSF CAREER award, and the Univerisity of Illinois College of Engineering's Rose Award and Everitt Award for Teaching Excellence. Prior to his work on education and computer architecture, he developed the first algorithm that allowed rendering arbitrary three-dimensional polygonal shapes for haptic interfaces (force-feedback human-computer interfaces). He holds 6 patents.

Mr. Robert Timothy Deloatch, University of Illinois Urbana Champaign
Jacob Bailey, University of Illinois

Jacob Bailey is currently a sophomore studying computer science at the University of Illinois.

Bhuwan B Khattar
Dr. Wade Fagen, University of Illinois, Urbana-Champaign

Dr. Wade Fagen is a Lecturer in the Department of Computer Science in the College of Engineering at The University of Illinois at Urbana-Champaign (UIUC). He teaches one of UIUC's largest courses, Introduction to Computer Science, known as CS 105. His research aims to improve learning by using technologies that students already bring to the classroom.

Dr. Cinda Heeren, University of Illinois, Urbana-Champaign

Dr. Cinda Heeren is an award-winning Senior Lecturer at the University of Illinois, Urbana-Champaign. She teaches CS225, Data Structures and Programming Principles, to hundreds of enthusiastic and talented undergraduates every year. She is always game to try new pedagogical innovations, and she loves telling young women about her affection for computing.

David Mussulman, Engineering IT Shared Services, University of Illinois at Urbana-Champaign
Prof. Matthew West, University of Illinois, Urbana-Champaign

Matthew West is an Associate Professor in the Department of Mechanical Science and Engineering at the University of Illinois at Urbana-Champaign. Prior to joining Illinois he was on the faculties of the Department of Aeronautics and Astronautics at Stanford University and the Department of Mathematics at the University of California, Davis. Prof. West holds a Ph.D. in Control and Dynamical Systems from the California Institute of Technology and a B.Sc. in Pure and Applied Mathematics from the University of Western Australia. His research is in the field of scientific computing and numerical analysis, where he works on computational algorithms for simulating complex stochastic systems such as atmospheric aerosols and feedback control. Prof. West is the recipient of the NSF CAREER award and is a University of Illinois Distinguished Teacher-Scholar and College of Engineering Education Innovation Fellow.

©American Society for Engineering Education, 2015

Computerized Testing: A Vision and Initial Experiences

Abstract

In this paper, we describe our experiences with a prototype computerized testing lab and running the bulk of a 200-student computer organization class's exams using computerized testing. We discuss the mechanics of operating the testing lab, the work required by the instructor to enable this approach (*e.g.*, generating a diversity of equivalent difficulty problems), and the student response, which has been strongly positive: 75% prefer computerized testing, 12% prefer traditional written exams, and 13% had no preference.

1 Introduction

In many college courses, exams contribute heavily to final course grades. As such, it is important that exams be an accurate and fair measurement of a student's understanding/ability, but exams are subject to resource and real-world constraints and involve tensions between multiple goals.

In particular, in large courses (*e.g.*, 200+ students), running exams can be a logistical nightmare. In many large courses, a scenario similar to the following occurs:

1. The hassle of running exams means that only a few (2-3) exams are run during the semester.
2. Exams are scheduled outside of regular class meetings to reduce the time pressure and enable alternate desk seating to make cheating harder.
3. Some students invariably have a conflict with the chosen exam time (or get sick), necessitating a conflict exam. Very large courses frequently have multiple such conflict exams.
4. To reduce the time it takes to return exams to students, as well as reduce grading effort, multiple-choice questions are used heavily.

With conflict exams, additional space and proctors need to be arranged, and either an additional conflict exam needs to be written or we risk student cheating resulting from communication between the different exam times. Furthermore, it can be difficult to write multiple-choice questions that address the higher levels of Bloom's taxonomy.[2]

In this paper, we pursue a different vision, enabled by making a computer a central part of the testing process. In Section 2, we describe our vision for a computerized testing center and the

perceived benefits of such an approach. In Section 3, we detail our experiences with running a series of 7 exams in the computerized testing center prototype.

2 Vision for a computerized testing facility

To reduce the logistical effort of running exams for each course, our vision is to centralize the effort of exam administration so that each course only needs to handle the content creation aspects of exams. The central piece of this vision is to have a dedicated exam center, which would be proctored a significant number of hours (*e.g.*, 60-80) a week. The usage flow of the testing center is roughly as follows:

1. When a course assigns a computerized exam, the professor specifies a range of days during which an exam can be taken. By specifying a range of days, we are able tolerate the fact that individual students can have athletic or other excused activities that could prevent them taking an exam on a particular day. Furthermore, the longer time period provides more student flexibility, which is good both for students and for fully utilizing the testing facility. The drawback of a longer exam period is that it delays returning exam results to students, if they are not provided immediately at the end of the exam (see Section 3.3).
2. For each computerized exam, the student reserves a time of their convenience from an online reservation system.
3. The student arrives at the testing facility at their arranged time. A proctor verifies the student's identity and the student is directed to a workstation; controlling student seat assignments makes it more difficult for students to coordinate cheating. The student's assigned machine has been booted into the specified exam configuration (many different exams are being run in the testing center concurrently). The student logs in to the machine using their campus computing password.
4. The student is presented with a series of questions to complete. Each student exam is different, consisting of a random selection drawn from a large collection of parameterized problems that together meet coverage and difficulty criteria.
5. The system displays the time remaining for the exam; when the time expires, the exam no longer accepts input from the student. In addition, the networking on the machine is configured to prevent access to unauthorized material and communication between students.
6. The student exam is archived and the automatically scorable portions of the exam are scored at this time. The exams and the scores are available to the faculty member any time after the completes.
7. Optionally, for the auto-graded portions of the exam, the student can be shown their score immediately after the exam along with the problems that they got wrong and their solutions.

2.1 Perceived advantages

We see two main advantages of this approach. First, it greatly reduces the faculty/course staff effort of running exams. Faculty can choose to offer an increased number of shorter exams, which

has been shown to have a number of potential benefits, including:[4]

- improving student class attendance.
- improving student satisfaction with the course.
- improving student learning (as measured by cumulative final exam scores) when item overlap exists between the shorter exams to exploit the spacing effect.[1]
- the largest gains coming from mid- to low-performing students.[3]

Furthermore, the low cost of exams allows them to be offered to students multiple times. "Second chance testing" allows students to replace part or all of an exam score by taking a second equivalent exam at a later date. Such an approach recognizes that traditional "one-shot grading" is much more effective at measuring a student's meta-cognition ability to recognize when they've sufficiently prepared for an exam than their ability or willingness to learn.[5] Alternatively, students can be required to repeat a test until particular core concepts are mastered in order to pass a course, to ensure pre-requisite chains are enforced.

Second, the use of a computer greatly broadens the kinds of questions that can be automatically graded. Most large courses rely on scantrons for automated scoring, which constrain questions to offer specific discrete answers from which students must select. With a computer involved, we can accept answers from students in a variety of formats: numerical values, equations, vectors drawn using the mouse, and even designs from simple CAD tools. One is limited only by one's creativity and what one can write a program to automatically grade. In Figure 1, we show a few example problems demonstrating some of the problems that we've created that would be challenging to ask in a traditional scantron format.

Furthermore, for computer science, we find computer-based exams to be particularly compelling for programming and systems-oriented courses, because students can be asked to write code in an environment where they can compile, test, and debug their code. While being able to write code on paper without the support of tools is a good thing, it isn't an authentic way that modern programmers work. We're much more concerned with our students ability to produce tests and use debuggers to make sure their code is 100% working. In Section 3.3, we'll show that students prefer this as well.

Beyond CS, many engineering disciplines are practiced in a heavily computer-supported environments, and computerized tests enable us to give students tests that involve students using industry standard software (*e.g.*, AutoCAD, Cadence, LabView, Pro-Engineer, SolidWorks) to solve design and analysis problems.

2.2 Relationship to Previous Work

Computer-based testing has had a long history and has been a significant focus of research, especially because it enables Computer Adaptive Testing (CAT).[8,9] Nevertheless, computer-based testing has largely only made had significant penetration in standardized testing (e.g., scholastic achievement testing, certification testing); its penetration into traditional post-secondary testing has been modest. Furthermore, the potential for non-multiple choice items has been previously

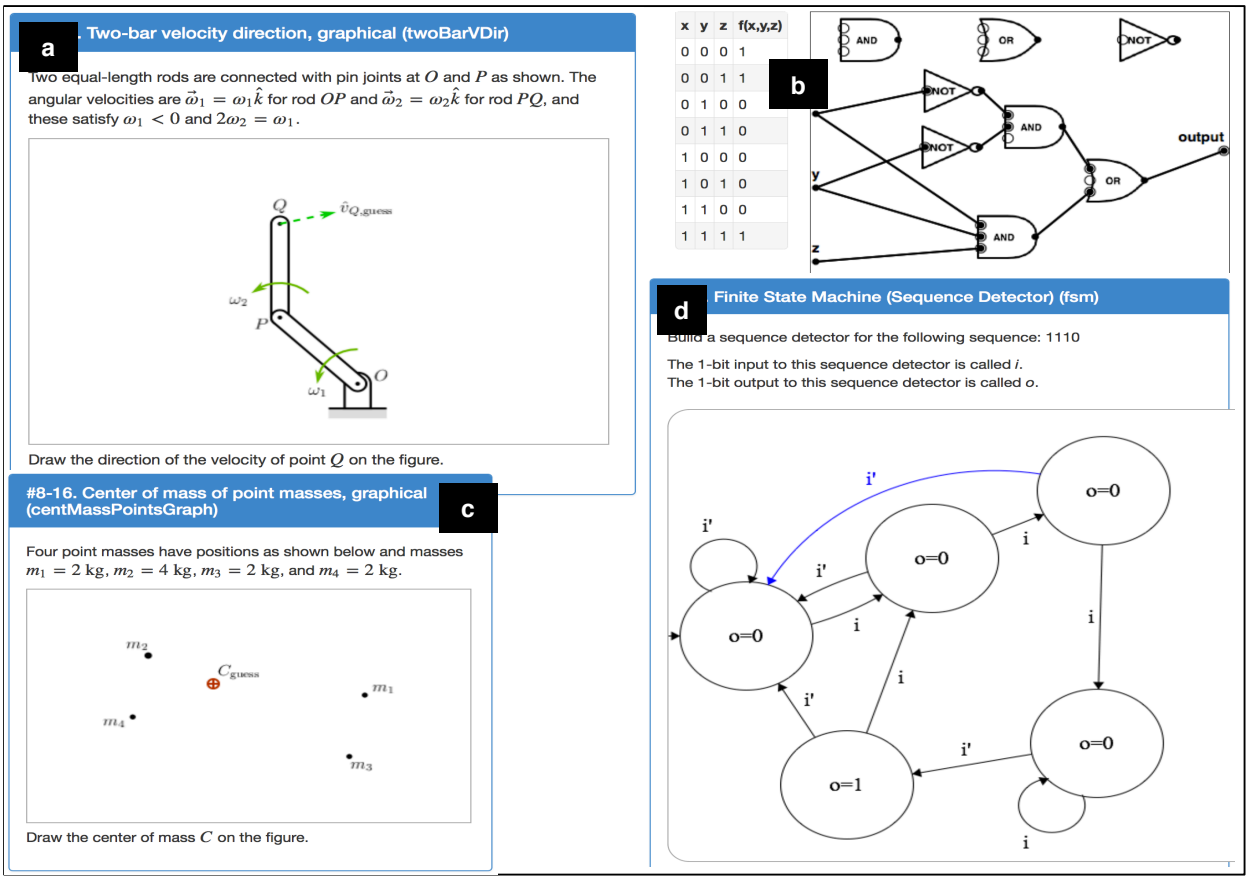

Figure 1: Example computerized problems showing more interesting behavior than just multiple choice: a) a graphical problem where the student needs to point the dashed arrow in the correct direction, b) a circuit design problem where the student needs to assemble the correct circuit, c) another graphical problem where the student needs to estimate the (x, y) position of the center of mass, d) a finite-state machine design problem where the student needs to place nodes and connect them with arcs.

recognized[6], but most computerized testing systems predominantly use/support items where the subject selects from a list of provided responses.

With the growth of online instruction, online testing (often mediated by a computer) has received attention in the literature as well. Two key issues that Rovai[7] identifies with online testing are identity security and cheating prevention. These issues are not a problem for our computerized testing approach, because we use proctored spaces that can verify the student's identity.

3 Experience with a prototype computerized testing facility

Over the summer of 2014, we set up a prototype computerized testing facility, as shown in Figure 2. The testing lab consists of 49 student workstations. Each display is outfitted with a privacy screen that prevents test takers from reading off the screens of neighboring computers. In addition, we have a second, much smaller room that holds only 4 workstations that is used to provide a "reduced distraction environment" for students registered through our Disability

Figure 2: Prototype computerized testing center during an exam. This basement room that holds 49 workstations has no windows and easily control ingress/egress. Note that monitors on the right of this image appear black because of the viewing angle on the privacy screens.

Resources & Educational Services (DRES).

When used for the testing center, the machines are booted into a Linux configuration we refer to as "exam mode". This configuration is mostly a normal Linux distribution with access to the same software packages that the normal campus Linux machines can access, but with a few important differences:

1. a distinct filesystem for home directories is used so that students can't access their normal home directory during the exam or access their exam home directory outside of the exam.
2. networking is limited (using `iptables`) to a few specifically white-listed IP addresses, to prevent students from accessing unauthorized materials and communicating with anyone electronically.
3. non-home directory data is wiped on login, so students can't leave files on the machine (*e.g.*, in `/tmp`) for subsequent students to read.

For the Fall 2014 semester, we used the computerized testing center for two courses. The most usage was in a 200-student computer organization course that ran computerized exams for 3 exams, their corresponding second chance exams, and a final. A second special topics course focusing on web programming used it for a single 20-person exam covering HTML, CSS, and JavaScript. In this paper, we will focus on the experiences from the computer organization course since it represents the bulk of the usage; for the single web programming exam, the students were given a collection of small self-contained JavaScript programming challenges as well as a single larger client/server web application that they were required to extend with new functionality.

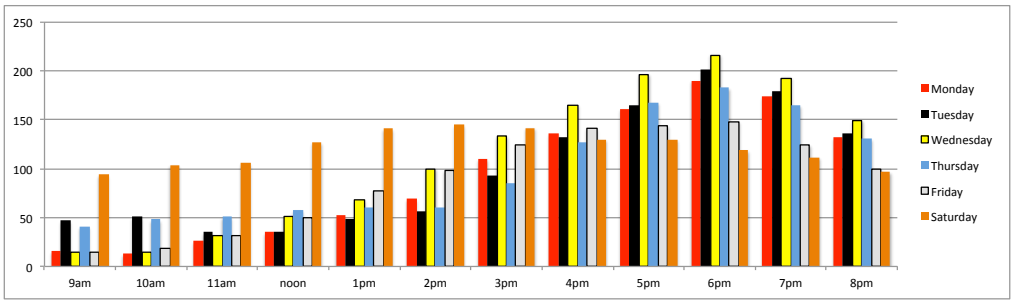

Figure 3: Student stated preferred/available exam times. Preferred times are largely after 4pm on weekdays and Saturday afternoons.

3.1 Student exam time preferences

Since we were primarily running exams for only one course, we didn't staff the testing lab very many hours/week. Given that we had 200 students and 49 computers in the main room, we chose to run 8 seatings for each exam. We provided much more capacity than we needed during this first run for two reasons: 1) it helps ensure that every student can register for an exam that doesn't conflict with their other obligations, and 2) it meant, as we'll show shortly, that our first offerings were sparsely attended, which allowed us to catch problems before running the exam with a large group of students.

To figure out when to schedule these exams, we solicited time preferences from students. We gave students an electronic form to fill out where they were provided times from 9am to 8pm on Monday through Saturday. Students could rank each time as 2 (a preferred time), 1 (an available, but not preferred time), or 0 (a not available time). Each student had to label at least 24 time slots with non-zero values. The resulting sums of the student responses is shown in Figure 3, which show a preference for weekday late afternoon/early evenings and Saturday afternoons.

Based on these student responses, we selected 8 times trying to ensure that every student could attend at least 2 or 3 times. Because our discussion sections/labs on Mondays and Tuesday, we chose to hold our exams from Wednesday through Saturday. We settled on the following times: Wednesday 3pm, 6pm, Thursday 5pm, 7pm, Friday 1pm, 6pm, and Saturday noon, 2pm. Figure 4 shows the times of the exams that the students actually signed up for. Clearly sign-ups are not uniform throughout the exam period, but rather could be modeled as a trapezoidal distribution. This suggests that when scheduling overlapping exams (from different courses), it is preferable to stagger the last day of the exam period so as to try to distribute the load and avoid creating days that are hot spots.

3.2 Exam Content

Key to the computerized exam approach is generating many versions of questions and assembling an individualized student exam from these questions. During our first semester of computerized exams, we used two different approaches to distributing problems to students – PrairieLearn and the filesystem – based on the complexity of the question.

For short answer questions, we used PrairieLearn [10,11], a web-based homework infrastructure

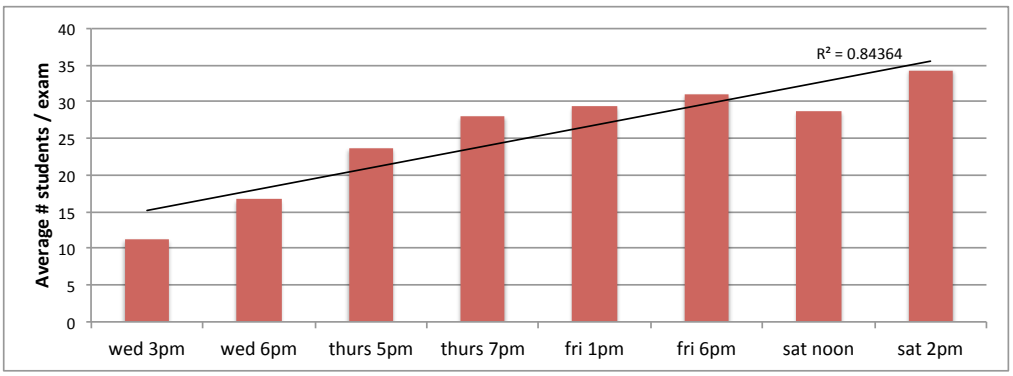

Figure 4: Distribution of times that students select to take exams. Students prefer to take exam at end of exam period.

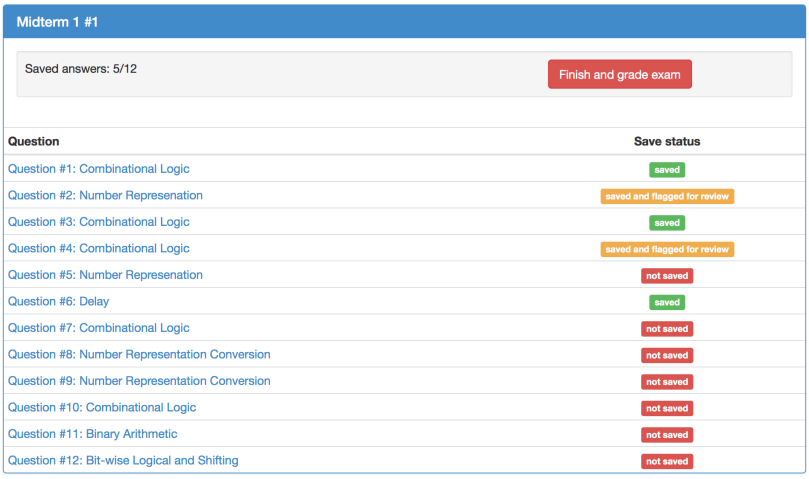

Figure 5: PrairieLearn interface for navigating through short answer questions.

designed to serve parameterized problems (including the ones shown in Figure 1) either as homework or exams. For exams, we set up a separate exam-only PrairieLearn server that only responds to client requests from the machines in the computerized testing center. PrairieLearn supports generating random exams by selecting problems from user-defined categories and generating a random instance of those problems. The problems are provided to the students through an interface that allows the students to sequence through the problems in any order, answering them in any order, marking some problems for later review, and finally submitting the problems for grading, as shown in Figure 5.

For these PrairieLearn questions, we grade the student's exam and show the results as soon as the student submits the exam for grading. We did this because it greatly simplifies the work flow, as it means that we don't need a way for the students to look at their PrairieLearn components outside of the testing center, which would require extra engineering since the exam PrairieLearn server doesn't support connections from outside of the testing center. We were initially concerned that students would react negatively to this immediate feedback, but this turns out to be an aspect of the testing center that students particularly like (Section 3.3).

For longer programming questions, we used the exam mode filesystem. Before an exam, we'd copy each student's programming problem into their home directory. Typically, this would consist of a prompt indicating what code should be written, some provided code and/or testing framework for use by the student, and a file intended for the students to edit to complete the assignment. After the student's exam time is over, we can read the files out of their directory and grade them. In all cases, we gave students example questions in the provided structure before the exam, so that they would be comfortable with the interface and could concentrate on solving the problems.

When we had many versions of a given programming question, we typically assigned problems to students randomly using their user name to create a random seed. In a few instances, we had sufficiently few question versions that we used one instance per seating, so that only students taking the exam concurrently would have the same question.

Because we needed to have many more versions of exam questions for the computerized exams relative to a conventional exam, it was a lot more effort up front to generate the first set of exam questions and the supporting infrastructure. It should, however, be significantly less effort going forward. By generating many versions of questions, we are trying to reach the point where, *even if students have all of our problems and their solutions*, it would be less effort for them to learn the course material than to memorize all of the solutions. Once we've reached that point, then we can re-use exam questions from semester to semester. More precisely, we plan to take our current pool of questions, periodically add new questions to the pool, and each semester use a subset of the questions.

Building up a pool of short answer questions for PrairieLearn, was relatively straight-forward for us for Fall 2014. We had already been using a web-based homework system for a number of semesters that included problem generators that randomized problem parameters to give students different problems. We ported a number of those questions into the PrairieLearn framework. Using these questions for both homework and exams actually saved us time relative to having to write separate new multiple choice problems for exams. In addition, we ported a number of old multiple choice questions directly into non-parameterized multiple choice questions in PrairieLearn.

Writing new parameterized questions for PrairieLearn depends on the complexity and how similar the question is to questions that have already been written; our experience is that writing a question from scratch can take as little as a half of an hour (if it is similar to question that has already been implemented) and as much as twenty hours (or more) for a question that requires significant novel graphics and/or non-trivial automatic grading routines implemented. Templates exist for non-parameterized questions (*e.g.*, multiple choice), which allow them to be implemented in as little as a minute if the source question already exists.

We found it significantly harder to write generators for the longer-form programming questions. Often the changes we wanted to make between questions (*e.g.*, an array of pointers vs. a linked-data structure) would have such structural impacts on the question that there wasn't much similarity left. As such, for most of these questions, we instead wrote distinct questions because doing so was less effort than writing question generators that gave meaningful diversity. As such, this first semester, we wrote 8-15 different versions of our long-form exam questions, which took

roughly 5-10 times the effort of writing a normal long-form exam question. It isn't quite twice the effort to write twice as many questions.

3.3 Student Feedback

We collected feedback from students in two ways: short open-ended surveys and informal post-exam interviews. The open-ended surveys asked two questions:

1. How do you compare computerized exams to traditional written (paper) exams?
2. What comments/concerns do you have about the computerized exams?

101 students (about 50% of the enrolled students) completed the surveys, but not every student answered both questions. Based on the responses, we make the following observations:

- The respondents generally preferred computerized exams, with 57 indicating a preference for computerized exams, 9 indicating a preference for traditional exams, and 10 indicating no preference.
- The biggest advantage of computerized exams appears to be that it gives students the opportunity to use a compiler (noted by 15 students) to catch syntax errors and to test their code (noted by 16 students). This prevents a number of small errors that usually cause students to lose points.
- Surprising to us, 9 students (without prompting) indicated that they liked the immediate feedback from PrairieLearn, which allowed them to see their scores and which questions they got wrong.
- The significant preference for computerized exams might be in part because our course is predominantly computer science majors; 8 students used words like "comfortable" and "less stressful" for computerized exams (perhaps because our students spend so much time on computers and in computer labs), while only 3 students identified the computerized exams as being "more stressful". Also, 3 students indicated that they preferred typing to writing, which is also perhaps unique to CS majors.
- The biggest concern, exhibited by 8 students on the survey, was how finicky some PrairieLearn questions were about the format of the responses they accepted. We believe this to be a shortcoming of specific questions rather than the approach; we're in the process of fixing those questions for the upcoming semester.

The post-exam interviews had many similar and complementary findings:

- Students found the reservation process easy and straightforward.
- Students found that using PrairieLearn for both homework and practice exams made worked well to prepare them for the interface and set expectations for difficulty of the exam.
- The PrairieLearn exam interface led the students to do the exam in a more linear fashion (even though this isn't required) than they would on a paper exam, because it was harder to skim in electronic format.

- Being able to test their code (for long-form programming questions) greatly improved their confidence in the correctness of their solution and reduced time they spent answering the question because they didn't feel compelled to double or triple check their code for errors.
- There was a lack of consensus about the fairness of exams where students receive different questions. Some students noted that some instances of the parameterized PrairieLearn problems were easier than others, and we overheard some discussion of students trying to compare difficult of exams after the exam was over. Other students believed that we only "changed the numbers" and each student is responsible for what is on their exam. Overall, we were surprised of how small of a concern this was for students; on the surveys, only 2 students commented negatively about fairness and 1 student commented positively about it.
- While students greatly preferred doing the programming questions on a computer, doing the short answer questions on the computer (through PrairieLearn) was primarily viewed as a concession to the course staff since it reduces grading time. Often students would still need to work out problems on paper and then enter the final answer into PrairieLearn.

3.4 Limitations of computerized exams

While we generally feel that having the full power of a computer at the student's disposal during an exam is generally an advantage (at least for running computer science exams), it can affect the kinds of questions that one can reasonably ask, at least with our current infrastructure. Specifically, it prevents asking questions that can be automatically answered with any of the tools accessible on the computer.

We first identified this problem during one of our exams. In our first exam, we included problems that require a student to translate numbers from decimal to binary representation (and vice versa). One student asked us if it was permissable to use the calculator program on the computer. We granted permission, as we didn't have a way to prevent or monitor other students usage in the exam timeframe. We didn't observe significant use of the calculator throughout the rest of the exams (most of the problems are simple enough to do in one's head or quickly on a piece of paper), it does bring up an interesting pedagogical question: what kinds of problems are we satisfied with students solving with the aid of software vs. what problems are important to solve without any external support.

For example, commonly in introductory programming courses, we ask students to answer questions about programming language syntax (*e.g.*, identify and correct all of the syntax errors in the following block of code) or ask students to execute a piece of code in their head and give us the result. These tasks are important to be able to do in one's head in the same way that automaticity of math facts (*e.g.*, multiplication tables) is crucial for students being able to do higher order thinking in math. Nevertheless, a computerized exam (in our current form) allows access to a source language compiler, which greatly facilitates solving these questions by highlighting syntax errors and allowing one to execute arbitrary code. In some instances a paper-based exam may be superior or a more locked-down computer environment that doesn't provide access to undesired tools.

Conclusion

In our first attempt to convert a large class to using all computerized exams, we were frequently building content and infrastructure at the same time. This had two consequences. First, a number of our exams ended up being later (and graded later) in the semester than they were in the previous semester. Second, sometimes things weren't as well tested when they went into production as they should have been, which led to some difficulties. Nevertheless, we are optimistic about computerized exams and the advantages they offer. Most of the problems we observed can be fixed by more careful engineering.

Our future work is on three main fronts. First, we intend to fix the shortcomings identified in the first offering of CS 233 in the computerized testing lab, for the offering in Spring 2015 to 350 students. Second, we're working to bring additional courses to the testing center, which requires some re-engineering of the systems that support it to allow people not involved in running the testing center to use it. Third, we'd like to use the testing center as a research test bed to better understand how computerized exams should be performed, so as to yield the best experience for students and faculty.

Acknowledgements

We're grateful to the College of Engineering at the University of Illinois and the Strategic Investment in Instruction Program (SIIP) for providing space and funding for the prototype computerized testing center. We'd like to thank Joe Zalabak and the Engineering IT instructional support group for setting up the testing center and all of their logistical and technical support. We'd also like to thank Tim Stelzer for conversations that helped lead to and helped shape this effort.

References

[1] J. L. Azorlosa and C. H. Renner. The effect of announced quizzes on exam performance. *Journal of Instructional Psychology*, 33:278–83, 2006.

[2] B. S. Bloom and D. R. Krathwohl. *Taxonomy of educational objectives book 1: Cognitive domain.* Addison Wesley Publishing Company, 1984.

[3] F. M. Kika, T. F. McLaughlin, and J. Dixon. Effects of frequent testing of secondary algebra students. *Journal of Educational Research*, 85:159–62, 1992.

[4] T. Kuo and A. Simon. How many tests do we really need? *College Teaching*, 57:156–160, 2009.

[5] C. E. Nelson. Student diversity requires different approaches to college teaching, even in math and science. *American Behavioral Scientist*, 40, Nov-Dec 1996.

[6] C. G. Parshall, T. Davey, and P. J. Pashley. Innovative item types for computerized testing. In WimJ. van der Linden and GeesA.W. Glas, editors, *Computerized Adaptive Testing: Theory and Practice*, pages 129–148. Springer Netherlands, 2000.

[7] Alfred P Rovai. Online and traditional assessments: what is the difference? *The Internet and Higher Education*, 3(3):141–151, 2000.

[8] Wim J van der Linden, Cees AW Glas, et al. *Computerized adaptive testing: Theory and practice*. Springer, 2000.

[9] Howard Wainer, Neil J Dorans, Ronald Flaugher, Bert F Green, and Robert J Mislevy. *Computerized adaptive testing: A primer*. Routledge, 2000.

[10] M. West. Prairielearn. https://github.com/PrairieLearn/PrairieLearn.

[11] M. West, C. Zilles, and G. Herman. Prairielearn: Mastery-based online problem solving with adaptive scoring and recommendations driven by machine learning. In *American Society for Engineering Education (ASEE) Annual Conference*, 2015.