BlueBook: A Computerized Replacement for Paper Tests in Computer Science

Chris Piech
Stanford University
Stanford, CA
cpiech@stanford.edu

Chris Gregg
Stanford University
Stanford, CA
cgregg@stanford.edu

ABSTRACT

This paper presents *BlueBook*, a lightweight, cross-platform, computer-based, open source examination environment that overcomes traditional hurdles with computerized testing for computer science courses. As opposed to paper exam testing, *BlueBook* allows students to type coding problems on their laptops in an environment similar to their normal programming routine (e.g., with syntax highlighting), but purposefully does not provide them the ability to compile and/or run their code. We seamlessly transitioned from paper exams to *BlueBook* and found that students appreciated the ability to type their responses. Additionally, we are just beginning to harness the benefits to grading by having student answers in digital form. In the paper, we discuss the pedagogical benefits and trade-offs to using a computerized exam format, and we argue that both the students and the graders benefit from it.

KEYWORDS

Computerized Exam; Assessment; Pedagogy

ACM Reference Format:
Chris Piech and Chris Gregg. 2018. BlueBook: A Computerized Replacement for Paper Tests in Computer Science. In *SIGCSE '18: SIGCSE '18: The 49th ACM Technical Symposium on Computer Science Education, February 21–24, 2018, Baltimore, MD, USA*. ACM, New York, NY, USA, 6 pages. https: //doi.org/10.1145/3159450.3159587

1 INTRODUCTION

Although it is not universal, most university introductory computer science courses give traditional paper exams, asking students to hand-write code for the problems that test programming ability. Pedagogically, there are legitimate reasons for having students hand-write code. However, hand-written code does have its downsides, including the tedious nature of undoing (hand-erasing) incorrect work, messy handwriting (leading to longer grading times), and space limitations on paper. More importantly, students who are used to typing code for their assignments are forced to write code in a completely different manner when taking paper-based exams, and this can be stressful and does not necessarily assess the students accurately. Additionally, grading handwritten exams cannot benefit from automatic grading tools, which can be tremendously helpful as course enrollments escalate. In this paper, we present *BlueBook*, a lightweight, cross-platform, computer-based, open source examination environment that students can run on their laptops. There is nothing particularly fancy about *BlueBook* (see Section 3), though it does include public-key password-protection (so the students can download exams), encrypted local backups, and the ability to automatically submit exams to a server with a historical log of student work during the exam. *BlueBook* also provides syntax highlighting for code writing, but we purposely decided not to include any ability for students to compile and/or test their code (see Section 5).

With BlueBook, we believe that we have finally reached a point in our courses where computerized testing is a better assessment tool than paper-based testing. Our computer science department experimented with a computerized testing framework roughly a decade ago and at that time we were disappointed with the results. This was primarily due to two reasons: scaling the testing to large classes was difficult, and allowing the students to compile and test their code inhibited students' ability to finish the exams.

The scaling issue was almost purely technology-limited. At that time, not all students had laptops, and battery-technology was also at a point where most laptops could not hold a charge for an entire 3-hour final exam. Therefore, the tests were administered in a computer lab on campus, and this presented logistical problems for large classes. Now, 100% of our students have laptops, and while there are still some battery issues, we have found that in a class of 300 students, approximately 10% needed access to mains power, which was easy to provide through the use of power strips and extension cords.

The other main reason we ended the original computerized testing experiment was because we found that students who were able to compile and test their code spent too much time on getting the code to work. They were not finishing the exam problems and were frustrated when they could not produce fully-working solutions. Although we would still like to explore a modified compile-and/or-test framework, for *BlueBook* we decided to only allow students rudimentary syntax highlighting, and we found that students were able to complete the exams on time. See Section 5 for a discussion on allowing students to compile their code.

In this paper, we demonstrate how *BlueBook* can be both beneficial to students and graders, and we present pedagogical arguments for the use of a computerized testing environment. We also present evidence of our successful transition away from paper-based exams, as well as some of the remaining challenges that can be expected from using this type of test-taking system.

2 RELATED WORK

Computer-based testing (especially in computer science) has a long history of research and experimentation. The main focus of much of the research has been regarding student performance: is there a

SIGCSE '18, February 21–24, 2018, Baltimore , MD, USA

ACM ISBN 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159587

quantitative difference in student performance between paper exams and computerized exams? Although this paper does not focus on that aspect particularly, we did not want to embark on our project without understanding the potential ramifications of student performance if we transitioned to computerized exams. Most of the research indicates that student performance is comparable between paper and computerized exams, and in many cases assessment was improved with the use of novel computerized tests. For example, in the early 1990s, Syang and Dale describe "Intelligent Tutoring Systems" that uses an adaptive test to assess performance[21], and adaptive testing is used on some prominent standardized tests, such as the SAT and the GRE[17, 24] We wanted *BlueBook* to model a paper-based exam as much as possible, though in Section 8 we discuss modifications that will diverge from this idea. There are also many reports on web-based testing[20, 26] and tablet-based testing[4, 23]. Because of the limited security available with web browsers, and because of the necessity for Internet access for web-based tests, we did not pursue a web-based model. We also wanted a test that was accessible by all of our students, and 100% of our students have laptop computers, so we decided to make it a Java-based program that could run on typical laptop computers.

Grissom et al.[8], and Lappalainen et al.[13] both reported that computer-based alternatives to paper tests gave students a better chance to fix errors, although in both experiments students were allowed to compile and test their code. As we discuss in Section 5, we purposefully did not allow students to compile or test their code, but this is a possible extension of the project.

Most of the research we have read indicates that there are no significant differences in performance between paper and computerized tests, particularly in early programming classes[3, 9, 12, 16, 19, 22, 23]. There are further studies that show distinct benefits to computer-based testing[11, 14, 25], and others that claim computerized testing has gender-equalization benefits[3, 25]. However, there are some reports that computerized testing can be detrimental, particularly when general access to computers (based on socioeconomic and cultural factors) was taken into account[15]. Because our students already use laptops on a daily basis, access is not an issue, but we think more research could be done to investigate this concern.

Another concern of ours was the student perception of a computerized test versus a paper-based test. Students are used to taking paper exams for many of their classes, but they have also taken many computerized standardized tests in their academic path, as well. We were also concerned about their anxiety regarding computerized exams. There have been a number of studies to address student perception and anxiety, and they predominantly show that students prefer computer-based exams[6, 7]. Some studies show mixed results, potentially based on gender and familiarity with computers in general[10], but others claim that there are benefits to taking a computerized exam for low-achieving students[18].

3 BLUEBOOK DETAILS

BlueBook is a Java program for administering exams. The program runs in full-screen mode (to inhibit students from attempting to use other programs, or the Internet), as can be seen in Figure 1. Students choose from among the problems listed at the top of the screen, and the question appears on the left side of the screen with a basic editor for answers on the right side of the screen. The questions can include both formatted text and graphics, and answers are typed into the editor. Code answers can be syntax-highlighted based on programming language. There is a count-down timer at the top of the screen, and this can be modified for different groups of students taking the exam (e.g., students who are allowed extra time can get an exam with a longer timer). *BlueBook* can be set up to either stop the test at the end of the timer, or it can simply act as a reminder.

Creating a *BlueBook* exam is straightforward, with problems written in basic HTML and answer starter code in plain text. Instructors run a separate Java program that creates the exam data with a given password, time limit, and other constants (e.g., a server address and directory location for results), and the exam data is then packaged with the *BlueBook* exam program. Students can download the exam data and *BlueBook* program prior to the exam. The exam data is encrypted with a public/private key encryption that would make it materially impossible for students to determine the questions before the exam time. Students do not need Internet access during the exam itself, though they submit the exams at the end via the Internet (or they can copy it to a flash drive for submission, if necessary).

When the exam is administered, students are given a password to begin the exam. *BlueBook* backs up student responses at an instructor-defined time interval (we have found that every 30 seconds provides adequate granularity), and the entire history of student answers is uploaded to the server (see Section 8, which discusses using this historical data). The backup is also encrypted with public key encryption.

During the exam, *BlueBook* keeps the computer screen maximized and does not allow students to exit out of the program without closing it. We have also disabled switching to other programs, and *BlueBook* is the only program that can run during the exam. That said, there are potential ways for students to break the system (e.g., with a Virtual Machine), but we have done what we can to mitigate the potential for cheating.

When students finish the exam, they can submit the exam to a server automatically, which utilizes the `scp` protocol. Students do need Internet access for this step, but in a class of 400+ students, we did not find any issues with submissions. If students can't submit online, they can email the solutions, or provide them to us on a flash drive. Students receive an email receipt for their submission and the encrypted history is also kept on their laptops in the event that the submission did not get properly saved on the server.

If *BlueBook* crashes while a student is taking the exam, there is also a *Crash Recovery* program that can be used to retrieve the encryped answers and return the student to the exam. This requires an additional password that instructors or TAs type in (so the student cannot recover the data on his or her own).

Once the answers are received on the server, instructors can extract the data to either `json` or plain text format for grading. We have created an additional utility that produces PDF files for uploading into a grading program (e.g., Gradescope[1]).

As *BlueBook* is open source, features can be added or modified as necessary, and we hope that other developers add interesting features as the program matures. Please email the authors for access to the current source code.

4 PRACTICAL CONSIDERATIONS

In this section we discuss the practical reasons we have decided to transition to *BlueBook* from a paper-based exam system. The initial inspiration for *BlueBook* arose from the simple desire to reduce the

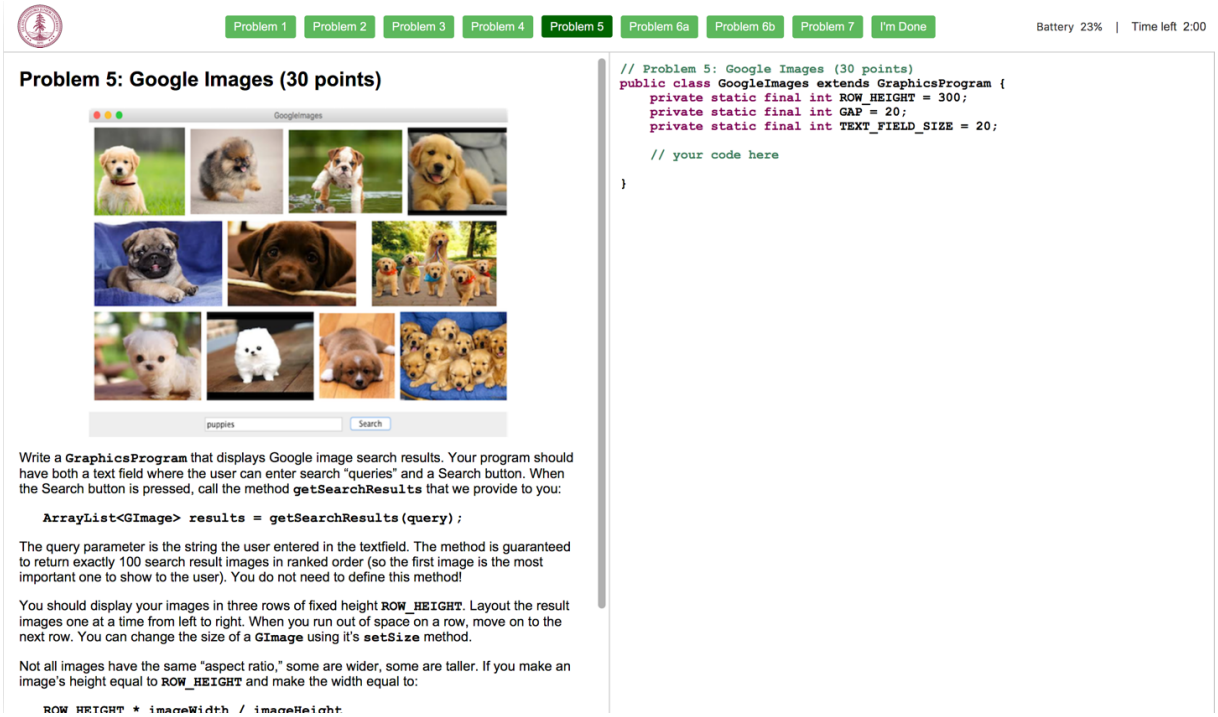

Figure 1: *BlueBook* Screenshot. The program runs in full screen mode, and prohibits students from switching between programs. Problems are stated on the left, and students answer on the right. Answers can be syntax-highlighted based on programming language.

amount of paper being used, and to minimize the logistical issues that using paper entails. Upon consideration of the ramifications of a digital exam, we quickly realized that there were numerous areas where we could benefit from the idea.

Our paper-based workflow included time consuming printing and scanning (we use an online grading system that accepts scanned PDF exams), and it seemed feasible to cut out the analog steps on both ends: if producing the exams is digital, and grading the exams is digital, we reasoned that it would be ideal to keep the exam experience digital, as well. For a 400 student class (which is about the average size of the introductory courses at Stanford), a 10-page, double-sided exam produces 2000 pages (four reams!) of paper. To physically produce an exam takes a significant amount of time, even if the printing and copying technology works perfectly. Once the exam is over, the scanning process involves removing staples (and keeping track of the loose pages of the exams), and running the exams through a scanner. Our department has an excellent scanner, but for large classes this process can take many hours, and getting the scanner to produce readable scans reliably often requires fine-tuning the scanner settings and some good luck.

The exam-taking process itself can also benefit from a computer-based exam. *BlueBook* is secure enough to allow students to download the encrypted questions ahead of time, so students can come into the exam, sit down, and start the exam immediately upon receiving the password. There is no need to hand out the exam, although we do allow students to use scratch paper, which is freely available around the exam room. Students do need to use their own laptops to take the exam, but we found that less than 10% of students needed access to power-points for their computers, and we were able to accommodate that even for very large classes. The timing of the exam is easier to manage, as well: students who show up a few minutes late will have their own timer on the exam and (if allowed) can simply take the extra minutes at the end of class to finish. This completely fixed the issue of trying to pry the exams from students at the end of the exam. Additionally, students who receive extra time on an exam can have their own data files that give them a timer appropriate for their accommodations. Because the exams are submitted wirelessly (and this is the only time students need Internet access), there is no need to physically collect anything at the end of the exam.

Because exam delivery is digital, we also found significant benefit for students who needed to take the exam remotely. Some of our classes have offsite students who take the exam with a local proctor, and *BlueBook* significantly improved the test-taking process for these students. The proctor simply needs to know the exam password to give to the students at the start of the exam. Other students who needed to take the exam remotely (e.g., traveling athletes) benefited similarly.

By and large, the students were happy with the practical benefits they received by using *BlueBook*. We heard many comments that the experience was better than taking a paper exam, and that it was more like their normal programming practice. See Section 6 for details on student experience.

Graders for *BlueBook* exams were unanimous in their support for using the program. Issues related to reading handwriting disappear with *BlueBook*, and grading typed text is altogether easier

than grading scanned, handwritten answers. An additional benefit derived from our ability to have students graded anonymously. In our pre-*BlueBook* grading, graders could use handwriting and a students name to infer either the individual or demographic details of the student they are grading. When grading digital exams we can avoid these unmeasured and undesirable problems.

We implemented an auto-grading system that allows graders to run student code during grading. This saved considerable time in grading students who had correctly or almost-correctly working code, and the only downside we noted was that virtually all student code starts with at least some syntax errors, and correcting those can take some time during grading. See Section 8 about ideas we are working on to mitigate this aspect of the auto-grading, and see Section 5 about why we don't allow students to compile and test their code during exams.

Practically, transitioning to a computer-based exam saves time and physical resources (e.g., paper and toner), students like it better than paper exams, and it opens up new grading avenues that are not available when the exams are analog.

5 PEDAGOGY

When we created *BlueBook*, we were understandably concerned about whether the idea was pedagogically sound. There is a long history of students handwriting code (both on exams and in general), and there are good arguments to suggest that students should demonstrate these skills while in a CS1 or CS2 course. Some of the arguments in favor of paper-based testing are listed below:

(1) By forcing students to write out their code, they will need to plan it out (whether in pseudocode or another form), and that planning is critical to being a good programmer.
(2) Many coding interviews require applicants to hand-write code (on paper or a white board, for instance), and handwriting code for exams gives students practice in these skills.
(3) When students take computer-based exams, they write too much code and by limiting the space on paper, they are forced to think through their code to create concise responses.

We do not necessarily find fault in the points listed above, but our opinion is that they are minor concerns that can be mitigated elsewhere in the computer science curriculum (or even in CS1 and CS2 courses in different assessments). Additionally, except in programming interviews, programmers rarely write code out by hand. We do think that it can actually be detrimental to students to have to hand-write programs when they are used to typing them, and during an exam we would rather students focus on solving the problem than potentially being distracted by a new way of writing their programs down.

When we brought up the idea to try a computerized test in our department, we learned that it had been tried before, with poor results. In the previous experiment, students were expected to write their answers and also had the ability to compile their code, and (most importantly) to run the code through a suite of tests. This proved disastrous, as students spent too much time trying to get their code to pass the tests, and many students did poorly on the overall exam because they were not willing to move to other problems if their code was failing tests. Partially because of this data, and particularly because we wanted *BlueBook* to mimic a paper testing environment as much as possible, we made the decision to disallow any compiling or running of code during the exam. Although occasionally students requested those features, the majority of students did not concern themselves with it, and seemed more pleased with the syntax highlighting and with the simple ability to type their answers. From a pedagogical standpoint, disallowing compiling or running of code does force students to think through their answers fully, and students can not simply try different solutions until they land on one that works (as is, unfortunately, what some students do on coding assignments).

Argument (3) discusses the limiting effect of a paper exam, to encourage students to write concise code that fits on the paper. We did not limit code size in *BlueBook*, and we did find that some students wrote more code than was required to answer some problems. We were initially concerned that this would be a bigger issue than it was, and it would be easy to modify *BlueBook* to limit student answer length. We also sometimes add in the problem description a length suggestion (e.g., "the reference solution is ten lines of code"), and this influences students to produce a similar length for their answers. Additionally, when limiting paper answer space, students with larger (or messier) handwriting are unfairly penalized, and this bias is eliminated with typewritten answers.

6 STUDENT EXPERIENCES AND FEEDBACK

When we first introduced *BlueBook* to one CS2 class, we gave the students a practice exam using *BlueBook* a few days before a normally scheduled midterm, which was to be available with both *BlueBook* and on paper (student's choice). We had roughly one hundred students take the practice exam, and we "exit polled" them after the exam to get their feedback and to provide *beta* test bug reports for the software. The feedback was predominantly positive, with over 95% of the students polled indicating that they would elect to take the exam using *BlueBook* for the actual midterm. The following represents a (paraphrased) sample of the specific responses to our questions:

- I liked the ability to type answers and to edit my responses.
- I didn't feel as constrained with space as on a paper exam.
- The countdown timer was helpful so I could pace myself.
- I was able to answer questions faster and better because I type faster than I write.
- My hand doesn't feel cramped like it does after paper exams.
- I liked having the question and answer on the same page, so I didn't have to flip back and forth.
- The syntax highlighting was helpful.
- We're saving the trees!

During the practice exam the program crashed for three students, but they were able to quickly get back to the exam with the help of a TA with the crash recovery password. No data was lost.

For the midterm and final exam for the initial CS2 class, roughly two-thirds of the students elected to take the exam using *BlueBook*. We allowed students the use of as much scratch paper as they wanted to use, and we provided power strips for those worried about battery life. The following is a sample of reasons students elected to take the paper-based exam:

- My laptop can't last for the entire exam [*note*: despite available power strips].
- I'm used to paper and it feels more normal.
- The font is too small on my computer [we have fixed that issue]

After the midterm exam, we polled the class about whether they wanted us to administer the final exam using *BlueBook*. There was an overwhelming desire from the students to allow them to use it, to the point that the course staff admitted that there would be a great number of upset students if we didn't have a *BlueBook* option.

One concern we have about the student experience is that laptops, by the nature of their vertical screens, are easily viewable by nearby students. We don't know if there was more cheating during *BlueBook* exams, but this is a downside to the format.

We have successfully used *BlueBook* during a subsequent CS2 offering, and we only allowed paper exams on a case-by-case basis. We gave students practice *BlueBook* exams (to take on their own), and we reminded them about having a full battery multiple times before the exams. Less than five percent of the students ended up taking the exam on paper (for similar reasons as above). We did not receive any direct complaints about using *BlueBook*, and students readily accepted the exams in that format.

7 GRADING BENEFITS

Although we did not initially anticipate it, we found that having digital answers was a boon for exam grading. As discussed in Section 4, we saved a considerable amount of time post-exam with no need to shuffle vast amounts of paper for scanning into the online grading program. For the final exam in the most recent course to use *BlueBook*, we scheduled the grading for three hours after the exam, and we could have cut that time down if necessary. We do have to post-process the exams for the auto-grader and for uploading to the online grading program, but we have scripted this to streamline it.

Graders were happy that they did not have to struggle to read handwritten code, and we also realized that we likely reduced grading bias that might happen with neat-vs-messy handwriting.

We wrote an online auto-grader for student code that is still in its preliminary stages but that shows great promise (see Section 8 for our anticipated future work). All code problems from *BlueBook* get uploaded to an online database that is available to graders, and a web-based front-end allows graders to compile and run student code in a testing framework. As with any auto-grading, the specific code tests do have to be prepared separately, and this is not necessarily trivial. However, we found that creating the grading tests before the exam provided a nice forcing function to ensure that the exam questions were reasonable and that the rubric solutions worked correctly.

Interestingly, very few student responses compile straightaway, but the graders quickly get used to fixing syntax errors to get the code to compile. The workflow for most graders was as follows:

(1) Scan the student response in the grading software.
(2) Switch to the auto-grader and attempt to fix syntax errors and test.
(3) If the testing showed correct results, immediately mark the answer as correct.
(4) If the testing showed incorrect results, revert to the grading rubric, fixing the code to test as needed.

Graders reported that having the auto-grader was normally helpful, and if it did not always save time, it gave the them additional peace-of-mind that they were grading the problems better. We frequently found graders helping each other debug student code to enhance grading.

8 FUTURE WORK

We have a number of enhancements planned for *BlueBook* itself, for improving the grading after the exam has been processed, and also for research purposes. As we mentioned before, we also hope that by making *BlueBook* open source it can improve even faster.

8.1 BlueBook Enhancements

Compiling/Running Code Despite the past discouraging results of allowing students to compile and test their code, we would like *BlueBook* to have the option for one or both of those features. We think there is a the potential to allow students to at least compile (or perform *lint*-like behavior) so they can fix some of their own syntax errors (so graders do not have to do it). We are considering allowing students to compile only at the end of the exam for a certain amount of time, or only to compile a fixed number of times. As discussed in Section 5, allowing testing of code during an exam is riskier, but having the option would allow for interesting research about how students run code in a timed testing environment.

Live Exam Updating As every instructor knows, exams are rarely perfect, and in-class announcements about problems are sometimes necessary. In its current state, *BlueBook* does not require Internet access during examinations, but we would like to introduce an update functionality to allow for tests to be updated mid-exam for typos and clarifications. This could cut down on announcements, although there is a concern about how the news is delivered to the students (e.g., as a pop-up message?)

Increased Security We are confident that the current security for *BlueBook* is adequate, but there is a potential for cheating if students aren't taking the exam at the exact same time (e.g., for remote students). We plan on adding a feature that logs students when they first start the exam (which will require temporary Internet access), so that we can track when the exam was accessed.

Per-problem Timer Although students reported that the timer was beneficial to their pacing of the exam, we are considering adding more granularity in the form of an individual timer per problem that shows students how much time they have spent on a particular problem. This could be further expanded by providing students an "optimum response time," as described by Delen[5].

Graphics and Math Answers At the moment, *BlueBook* works well for plain-text and coding questions, but it is not appropriate for courses where students need to answer with drawings, or with mathematical symbols. Both of these features could be added. Mixing text with drawing provides a challenge for auto-grading, but that problem is not insurmountable. If we were to add a math/equations editor, students would necessarily need practice using it before it would make practical sense to use for an exam.

8.2 Better Grading

Auto-Syntax Correction Because we now have digital exam answers, we can use the information to better inform our grading. We would like to enhance the auto-grader in a couple of particular ways. The first is to attempt some basic

automatic syntax error correction. Automatic syntax error correction is a decades-old problem (see [2], for instance), but during grading we see regular syntax errors that we don't penalize students for (unless they are egregious), and we would like to investigate fixing them automatically. Very few student code answers compile immediately, but most can be fixed with a small number of easy-to-see corrections.

Nearest Correct Code Another area we have started implementing is to provide a "nearest correct code" window alongside code that needs to be graded. Any code that has been graded as correct will be flagged in the database, and as graders are looking at an ungraded problem, the closest correct solution will be shown in an adjoining window. We are working on a neural network that determines which of the correct solutions is the closest. We expect that such a feature will have the dual benefit of speeding up grading and also helping to normalize grades for similar solutions.

8.3 Research Opportunities

Because *BlueBook* captures student work at an instructor-defined frequency (we have set ours to collect responses every 30 seconds), there is a tremendous amount of data that can be mined regarding how students take examinations. For example, how long do students spend on individual problems, and do they spend more or less time on earlier problems? Do students routinely go back and make significant changes to early problems? Do students write code all at once, or do they make frequent pauses?

As mentioned above, there are potential research questions regarding allowing students to compile and/or run their code during examinations, and once we implement these features into *BlueBook*, we will be able to investigate them.

Finally, we would like to collect data on how graders are assessing students when they have the ability to run and test student code. With some of the grading updates mentioned above, we can start research in that direction.

9 SUMMARY

Our implementation and use of the *BlueBook* computerized exam software has had multiple and significant benefits to our CS2 course. It has helped streamline and improve our exam logistics both before, during, and after the exams. Students have embraced the software and have had provided predominantly positive feedback. Indeed, some students would have been disappointed if they were not allowed to use *BlueBook* on exams. Graders were happy to improve their experience with the ability to compile and test student responses. We likely minimized bias and saved time due to handwriting issues. We are excited about the research avenues we are planning based on *BlueBook*'s data gathering and reporting, and by future enhancements to the software.

ACKNOWLEDGMENTS

The authors would like to thank Stanford undergraduates Ali Malik and Brahm Capoor, who have spent countless hours helping with the coding and roll-out of BlueBook.

REFERENCES

[1] [n. d.]. Gradescope. https://gradescope.com. ([n. d.]). Accessed: 2017-08-24.
[2] Alfred V Aho and Thomas G Peterson. 1972. A minimum distance error-correcting parser for context-free languages. *SIAM J. Comput.* 1, 4 (1972), 305–312.
[3] Eren Can AYBEK and R Nükhet DEMİRTAŞLI. 2014. A Comparison of Psychometric Properties of a General Ability Test Which Administered In Paper-Pencil and Computer Based Form. *Elementary Education Online* 13, 4 (2014), 1400–1413.
[4] Matthew J Cheesman, Prasad Chunduri, Mary-Louise Manchadi, Kay Colthorpe, and Ben Matthews. 2015. Student Interaction with a Computer Tablet Exam Application Replicating the Traditional Paper Exam. *Mobile Computing* 4 (2015), 10–21.
[5] Erhan Delen. 2015. Enhancing a Computer-Based Testing Environment with Optimum Item Response Time. *EURASIA Journal of Mathematics, Science and Technology Education* 11, 6 (2015), 1457–1472.
[6] Robert Deloatch, Brian P. Bailey, and Alex Kirlik. 2016. Measuring Effects of Modality on Perceived Test Anxiety for Computer Programming Exams. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 291–296. https://doi.org/10.1145/2839509.2844604
[7] Victor Faniran and Nurudeen Ajayi. 2016. Students' perceptions of computer-based assessments: A case of UKZN. In *IST-Africa Week Conference, 2016*. IEEE, 1–9.
[8] Scott Grissom, Laurie Murphy, Renée McCauley, and Sue Fitzgerald. 2016. Paper vs. Computer-based Exams: A Study of Errors in Recursive Binary Tree Algorithms. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 6–11. https://doi.org/10.1145/2839509.2844587
[9] D. Gürsoy. 2016. Assessing novice programmers' performance in programming exams via computer-based test. (June 2016). http://essay.utwente.nl/70114/
[10] Pari Delir Haghighi, Judy Sheard, Chee-Kit Looi, David Jonassen, and Mitsuru Ikeda. 2005. Summative Computer Programming Assessment Using Both Paper and Computer.. In *ICCE*. 67–75.
[11] Monirosadat Hosseini and Seyyed Morteza Hashemi Toroujeni. [n. d.]. Replacing Paper-Based Testing with an Alternative for the Assessment of Iranian Undergraduate Students: Administration Mode Effect on Testing Performance. ([n. d.]).
[12] Hanho Jeong. 2014. A comparative study of scores on computer-based tests and paper-based tests. *Behaviour & Information Technology* 33, 4 (2014), 410–422.
[13] Vesa Lappalainen, Antti-Jussi Lakanen, and Harri Högmander. 2016. Paper-based vs Computer-based Exams in CS1. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research (Koli Calling '16)*. ACM, New York, NY, USA, 172–173. https://doi.org/10.1145/2999541.2999565
[14] Karen A Maguire, Daniel A Smith, Sara A Brallier, and Linda J Palm. 2010. Computer-based testing: A comparison of computer-based and paper-and-pencil assessment. *Academy of Educational Leadership Journal* 14, 4 (2010), 117.
[15] Óscar Marcenaro-Gutiérrez and Luis Alejandro López-Agudo. 2016. MIND THE GAP: ANALYSING THE FACTORS BEHIND THE GAP IN STUDENTSâĂŹPERFORMANCE BETWEEN PENCIL AND COMPUTER BASED ASSESSMENT METHODS. *Revista de Economía Aplicada* 24, 71 (2016).
[16] Alan D Mead and Fritz Drasgow. 1993. Equivalence of computerized and paper-and-pencil cognitive ability tests: A meta-analysis. *Psychological Bulletin* 114, 3 (1993), 449.
[17] Craig N Mills and Manfred Steffen. 2000. The GRE computer adaptive test: Operational issues. In *Computerized adaptive testing: Theory and practice*. Springer, 75–99.
[18] Stavros A Nikou and Anastasios A Economides. 2016. The impact of paper-based, computer-based and mobile-based self-assessment on students' science motivation and achievement. *Computers in Human Behavior* 55 (2016), 1241–1248.
[19] Jan M Noyes and Kate J Garland. 2008. Computer-vs. paper-based tasks: Are they equivalent? *Ergonomics* 51, 9 (2008), 1352–1375.
[20] Sanjay Kumar Singh and Arvind Kumar Tiwari. 2016. Design and Implementation of Secure Computer Based Examination System Based On B/S Structure. *International Journal of Applied Engineering Research* 11, 1 (2016), 312–318.
[21] Angel Syang and Nell B. Dale. 1993. Computerized Adaptive Testing in Computer Science: Assessing Student Programming Abilities. *SIGCSE Bull.* 25, 1 (March 1993), 53–56. https://doi.org/10.1145/169073.169109
[22] Krisztina Tóth. 2015. *THE COMPARATIVE ANALYSIS OF PAPER-AND-PENCIL AND COMPUTER-BASED INDUCTIVE REASONING, PROBLEM SOLVING AND READING COMPREHENSION TEST RESULTS OF UPPER ELEMENTARY SCHOOL STUDENTS*. Ph.D. Dissertation. szte.
[23] Andrew Valentine, Iouri Belski, and Margaret Hamilton. 2017. Developing creativity and problem-solving skills of engineering students: a comparison of web-and pen-and-paper-based approaches. *European Journal of Engineering Education* (2017), 1–21.
[24] Howard Wainer, Eric T Bradlow, and Zuru Du. 2000. Testlet response theory: An analog for the 3PL model useful in testlet-based adaptive testing. In *Computerized adaptive testing: Theory and practice*. Springer, 245–269.
[25] Patricia Wallace and Roy B Clariana. 2005. Gender differences in computer-administered versus paper-based tests. *International Journal of Instructional Media* 32, 2 (2005), 171.
[26] Yuan Zhenming, Zhang Liang, and Zhan Guohua. 2003. A novel web-based online examination system for computer science education. In *33rd ASEE/IEEE Frontiers in Education Conference*. 5–8.