Teaching A For-Credit CS1 Course as a MOOC

Cay S. Horstmann
San Jose State University
Cay.Horstmann@sjsu.edu

Abstract

In 2013, San Jose State University established a partnership with Udacity, a provider of massive open online courses. SJSU faculty and Udacity jointly convert college courses into the Udacity format and offer them both as for-credit courses and as free courses for the general public. In this article, I report on the challenges of authoring and delivering an introductory computer science course in this fashion, and conclude with recommendations for course design and vendor management.

Categories and Subject Descriptors K3.2 [*Computers and Education*]: Computer and Information Science Education

General Terms Human Factors, Design, Economics.

Keywords Massive open online courses, MOOC, CS1

1. Introduction

Massive open online courses (MOOCs), open-access courses that are offered to very large groups of students over the internet, have emerged about five years ago, and soon afterwards speculation began how they will transform the university and perhaps make it irrelevant. In 2013, San Jose State University (SJSU) and Udacity, a major MOOC provider, entered into an agreement to jointly develop and offer for-credit courses [1]. Three courses were offered in Spring 2013.

SJSU gave two principal reasons for the initiative: Lowering the cost of education, and increasing engagement with students. Registration for the MOOC version of a course is $150, and no textbooks are required. In contrast, a full-time in-state student pays about $550 for a three-credit course. The initial set of courses consisted of two remedial mathematics courses and a statistics course that traditionally had high dropout rates. SJSU expressed the hope that freshly desiged and professionally delivered online courses would be more attractive to students and increase success rates.

The computer science department decided to offer our introductory computer science course (which I will call CS1 in this paper) through Udacity. The course ran for the first time in Summer 2013. Like the other SJSU/Udacity courses, it does not replace the traditional course but offers students an alternative. Since our CS1 course regularly fills up and we have to turn away students, we welcomed the additional capacity that the MOOC course would provide. We hoped that the environment would enable us to offer more drill and practice than we normally do. Finally, we wanted to enable community colleges to teach our curriculum, using the free MOOC for student labs or a flipped classroom, so that transfer students can be better integrated into our program.

2. Our CS1 Course

We teach CS1 in Java, using an objects-early approach, covering the first 10 chapters of Big Java [2]. The course syllabus is essentially identical with that of the AP Computer Science A course [3]. About 2/3 of the students are currently CS majors or intend to join our program.

The objects-early approach is not without its distractors [4], but it has worked well for us. In particular, since the approach is new to all students, whether or not they had prior programming experience, it levels the playing field in the first few weeks. We use the BlueJ environment [5], and we introduce the object workbench as the primary tool for interacting with objects in the first three weeks, minimizing the dreaded `public static void main` method.

Our CS1 has a high failure rate (33.4% on average in the last six semesters), and instructors of the follow-on course complained that many passing students lacked essential programming skills. The latter was somewhat remedied in the last two years by changing the nature of the programming assignments. We used to assign four to six complex multi-week assignments, and changed to simpler weekly assignments. To handle the added grading volume, we use an autograder. We have human graders, but they focus on programming style, not correctness. The autograder enabled us to assign a draft version for each assignment, which asks for some partial work on the assignment, and is due four days before the final version. That draft, not surprisingly, yields many questions the day after its due date, when students are surprised by the challenges of the assignment. Having this surprise 12 times during the semester, with time to recover, seems to have been effective in increasing programming skills. But it dit not lower the failure rate.

We also have a very active discussion group. In Fall 2012, with about 150 students, our Piazza group had over 25,000 contributions. About half are answers to in-class clicker questions, but there is a very substantial discussion of the programming assignments. There is a small grade reward for participation. We also have a mandatory closed lab in which students become familiar with the development environment and the debugger, and where they practice developing algorithms and programs.

Udacity offers another CS1 course [8], but we felt that it was not comparable to our course. Not only does it use Python, but it is centered around the activity of building a search engine, and fails to cover a number of subjects from our syllabus.

3. MOOC Course Design

It was our intent to have the Udacity course mirror all the features of the regular course. At first glance, this seemed easy. The pub-

lisher provided a good set of slides, and since I had been using clicker questions in class, I thought that I could just replicate what I do in class: lecture for 15 minutes, ask a question or two, discuss the answers, and lecture again. Another MOOC provider, Coursera, offers quite a few courses in just that format, recording the whiteboard and showing a small inset of the lecturer's head, interrupted by an occasional multiple-choice question.

Udacity has a very different format. Each video segment is a minute or two long, and is followed by a question. The question can be a programming question, which is something that I had always wanted to do. (I had tried to ask programming questions in class, but it took a long time for students to complete even simple tasks, and setting up a way to get the results was tedious.) I was surprised how much the format changes the flow of the lecture. Consider for example the topic of method overriding. In my textbook, and in my lectures, I build up an example with a superclass and a subclass, tell students that they must call `super.method()` to avoid a recursive call, and that the subclass can't access the private implementation of the superclass. If you have the chance to ask a question every minute or two, you don't tell the student. You let the student build up the example, and you enable the student to experience what happens if one doesn't include `super` qualifier in the call to the superclass method.

The micro-lecture format also shines in the early lessons, when it perfectly meshes with the object workbench in BlueJ. I show off what an object can do, and then I ask the student to do something and tell me what happened.

Changing the material to this format was quite time-consuming, for three reasons. First, setting up the autograder requires both planning and testing. In fact, Udacity's autograder did not scale to the volume of questions in this course. (For example, the module on arrays has 29 such problems, where the equivalent module in Udacity's Python course has 11.) I convinced Udacity to use my autograder instead, which is optimized for quick problem setup.

Recording a video lecture is tedious because videos are so hard to edit. One wrong move, such as forgetting to turn on the camera, or the screencast software, or the microphone, can mean a wasted morning. Udacity's setup is particularly complex since they film the presenter's hand over a Wacom tablet and overlay the actual screencast. It took me several weeks of practice to master the tools and become comfortable enough with the workflow to be effective. I ended up modifying BlueJ, Xournal (a note-taking program for Linux), and OpenBox (the window manager) to automate repetitive steps, so that I could concentrate on my lecture without being distracted by menus, mouse clicks, and other menial tasks.

Finally, Udacity insisted that their non-paying audience was easily bored by traditional computer science subjects and that computing Fibonacci numbers or factoring an integer into primes were not viable activities. Since this audience was not captive, I was urged to come up with examples that were meaningful to young people. We developed a nice set of problems around a social network. There was much talk about integrating socially relevant topics, but most of them would have required an elaborate setup. In the end, I settled for visual interest and built a small library to support a media computation approach, similar to that advocated by Guzdial [6]. That library is optimized for use with the BlueJ workbench, refreshing images after every mutator invocation. Udacity also did a fine job producing interviews with computer science students, which were very motivating.

We collect the homework through the Udacity system and have it autograded. There is currently no human review, and we are evaluating whether that is reasonable by spot-checking student programs. Our format of frequent small homeworks is a good match for the capabilities of the system. We deem the exercises between videos to be the equivalent of the closed lab of the regular course.

Table 1. Pass and Retention Rates in the CS1 Course

Semester	Pass	Retention
Spring 2010	66.7%	94.3%
Fall 2010	61.5%	100.0%
Spring 2011	83.1%	100.0%
Fall 2011	82.0%	98.2%
Spring 2012	68.6%	95.9%
Fall 2012	69.2%	97.3%
Spring 2013	78.2%	99.2%
Summer 2013 (Udacity)	70.4%	61.0%

Overall, I spent about 100 hours recording (and re-recording) the lectures. Udacity provided a teaching assistant who presented about as much material as I did, saving me from another 100 recording hours. Setting up the programming problems took another 50 hours or so. Reviewing and fixing errors was very tedious and consumed a significant amount of time. It is astonishing how a massive audience will find every last error in every problem.

4. Student Performance

The pass rate of the online course was 70.4%, about the same as in the regular course, where SJSU reports it as an average of 67.6% over the last six semesters, with significant fluctuations from one semester to the next (see Table 1). However, we cannot conclude too much from that information. On the one hand, the composition of the Udacity class was very different from that of our typical classes. Many of the students are not SJSU students, and some are quite young, having been enrolled by their parents! Moreover, in the regular class, a student cannot drop after two weeks into the semester. We allowed Udacity students to drop until quite late. About 39% of students dropped.

Of the 478 remaining students, 50% got an A, a rate that is much higher than in our regular course (around 20%). We were wondering whether this is a consequence of the fact that students can keep submitting their answer to the autograder until they get it right (or they run out of time–deadlines were strictly enforced). However, last semester, we used the autograder in the regular course in the same way, and students did not achieve similar results. Tables 2 and 3 show the weekly homework outcomes in the Spring (regular) and Summer (Udacity) course. The percentages refer to the students that arestill enrolled in each week. It appears that the Udacity course further separates the well-known two modes in the grade distribution.

5. Relationship Issues

There were times when Udacity's business interests diverged from those of the university. Udacity has its own reasons to allow students to take the free course in parallel with the SJSU course. I already mentioned that they insisted that the course must be interesting for that group, but that's not a conflict, since it improves our course as well.

However, there was an issue about the difficulty of the programming problems. Originally, Udacity provided step-by-step pseudocode and expected students to translate it into Java. I was told that otherwise the free cohort would find the problems too frustrating. I got the problems changed, but it took some effort, and it raised the question who controls the content of a jointly offered course. In theory, SJSU has the power to set the content, but in practice, when one needs the vendor to set up the course in a very short timeframe, it is easy to get into sticky situations when they delay work that isn't in their interest.

Reporting was also a challenge. Students naturally expected that their grades would instantly appear in the LMS (the Canvas system

Table 2. Spring 2013 Homework Assignments

Assignment	A	F	Not turned in
0	88.36%	6.16%	5.48%
1 draft	73.29%	9.59%	9.59%
1 final	49.32%	8.90%	14.38%
2 draft	84.93%	4.11%	8.90%
2 final	77.40%	2.74%	6.16%
3 draft	51.37%	23.97%	8.90%
3 final	71.23%	6.85%	9.59%
4 draft	79.45%	4.11%	7.53%
4 final	80.82%	6.16%	7.53%
5 draft	84.25%	1.37%	8.90%
5 final	78.77%	6.85%	4.79%
6 draft	72.60%	4.79%	11.64%
6 final	54.11%	10.96%	8.90%
7 draft	72.60%	2.05%	14.38%
7 final	42.47%	12.33%	15.75%
8 draft	73.29%	4.79%	9.59%
8 final	70.55%	4.79%	13.70%
9 draft	77.40%	3.42%	13.01%
9 final	65.75%	7.53%	14.38%
10 draft	60.27%	9.59%	22.60%
10 final	69.18%	3.42%	17.12%
11 draft	68.49%	4.79%	16.44%
11 final	56.85%	10.27%	15.75%

Table 3. Summer 2013 Homework Assignments

Assignment	A	F	Not turned in
0	56.35%	1.90%	22.46%
1 draft	79.70%	6.09%	14.21%
1 final	83.42%	1.44%	13.58%
2 draft	69.19%	3.13%	24.41%
2 final	70.89%	3.13%	20.10%
3 draft	62.66%	1.83%	33.68%
3 final	62.92%	4.57%	28.46%
4 draft	54.73%	2.00%	39.68%
4 final	56.06%	4.53%	32.76%
5 draft	52.87%	2.14%	40.19%
5 final	56.10%	5.01%	32.38%
6 draft	57.10%	0.68%	40.16%
6 final	49.79%	6.31%	33.80%
7 draft	53.16%	2.10%	42.78%
7 final	54.26%	2.41%	38.35%

that we use at SJSU). Udacity has no support for scores or for FERPA-compliant grade reporting, and they were slow in providing the raw data to us, which caused significant student unrest. To get copies of the student work, for plagiarism checks and evaluation of the effectiveness of the autograder, was also frustratingly slow.

Exams are proctored by an outside service. Due to its fee structure, we were unable to give two 75 minute proctured midterms, as we do in the regular course, and had to settle for one 60 minute proctored midterm and one unproctured midterm, in addition to the proctored final exam.

As much as I like the short videos and student activities, I regretted that both Udacity and SJSU insisted that there be no textbook. The videos are not very useful as a reference, and many topics are quite boring to cover in a video. Students complained about the tedium of locating information in the videos. We created some ”fact sheets” to supplement the videos, but they were not very extensive.

Finally, the joint discussion forum was a problem. Our students were greatly outnumbered by participants in the free cohort, which Udacity reported at over 15,000. The SJSU instructor was not eager to support the public forum under these circumstances. By the time the course was completed, the forum had about 20,000 topics, many repetitive. Our students students expressed that it was overwhelming and not what they expected in a for-credit course. We ended up running a separate forum in our LMS just for them.

6. Conclusions

At SJSU, we don’t plan on replacing on-campus courses with MOOCs. However, MOOCS provide a valuable alternative, particularly for motivated students and in flipped classrooms.

It is too early to tell how the Udacity course will compare to our regular CS1 course, but initial observations are encouraging. There is little unhappiness in the discussion forum, and completion rate in the first homework assignment is only a little lower than in the regular course.

The core technique of repeated short videos, immediately followed by problems, is so compelling that I predict that we will see much more of it. Programming, calculus, French grammar, and many other topics benefit from constant computer-graded practice. There is no particular reason why this technique should be limited to MOOCs, and it is likely to appear in eBooks and learning management systems.I am not convinced that the videos are always the best mode of delivery. Repeated short readings, immediately followed by problems, might be just as effective.

When a university or a department wants to replicate what we have done, one option is to partner with an outside vendor such as Udacity or Coursera. An alternative is to use edX [7], a free and open-source MOOC platform. This raises the question what value an outside vendor provides.

In my experience, Udacity’s most valuable contribution has been the video production. It would be very difficult and costly to produce videos of comparable quality on campus. In contrast, the value of their software platform is less clear. Running an open-source platform, perhaps through a service provider, would give an institution quite a bit more control over content, reporting, and LMS integration.

Just like few faculty write their own textbooks, it is unlikely that many will want to produce high quality courseware. It is conceivable that textbook publishers will provide this material, to be plugged into the campus LMS or edX instance, or that MOOC vendors will morph into such publishers. For now, course creators have the uncomfortable choice between creating their own material, likely with limited visual appeal, or to partner with a vendor whose business interests are not in perfect alignment with the interests of the university.

References

[1] Harris, Pat L., SJSU and Udacity Partnership, Retrieved June 14 from the San Jose State University web site, http://blogs.sjsu.edu/today/2013/sjsu-and-udacity-partnership/

[2] Horstmann, Cay S. Big Java, 5th edition. John Wiley & Sons, 2013

[3] Computer Science A Course Description, The College Board, 2010

[4] Owen Astrachan, Kim Bruce, Elliot Koffman, Michael Klling, and Stuart Reges. 2005. Resolved: objects early has failed. SIGCSE Bull. 37, 1 (February 2005), 451-452.

[5] Kölling, M., Quig, B., Patterson, A. & Rosenburg, J. (2003) The BlueJ system and its pedagogy, Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, Vol 13, No 4.

[6] Guzdial, M. A media computation course for non-majors. In Proceedings of the 8th Conference on Innovation and Technology in Computer Science Education (ITiCSE), 2003.

[7] EdX builds community of developers for its online and blended learning platform, Press release, June 5, 2013, Retrieved June 14, 2013 from the edX web site, https://www.edx.org/alert/edx-builds-community-developers/944

[8] Mordechai (Moti) Ben-Ari. 2013. MOOCs on introductory programming: a travelogue. ACM Inroads 4, 2 (June 2013), 58-61.