SPOC-supported Introduction to Programming

Marco Piccioni
ETH Zurich
marco.piccioni@inf.ethz.ch

Christian Estler
ETH Zurich
christian.estler@inf.ethz.ch

Bertrand Meyer
ETH Zurich
bertrand.meyer@inf.ethz.ch

ABSTRACT

MOOCs (Massive Open Online Courses), which have taken higher education by storm, are an opportunity to elevate the quality of existing residential courses. We report about an experimental attempt during the Autumn 2013 semester at ETH Zurich, involving our "Introduction to Programming" course. We designed and implemented a MOOC infrastructure and used it as a SPOC (Small Private Online Course) to support and complement the existing course. The results reported in this article are encouraging for two reasons: first, the participation level was good, in spite of the fact that the online course was an optional addition to the residential course; second, students really liked the assessments (quizzes and programming exercises), in spite of the fact that assessments did not count towards the course final grade. The data we collected suggest that this may, at least in part, be due to a gamification aspect we introduced in the course: awarding virtual badges for obtaining full points in the quizzes.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer science education*

General Terms

Design, Human Factors

Keywords

CS1, MOOC, Pedagogy, SPOC

1. INTRODUCTION

The paper describes our experience with the Autumn 2013 edition of the "Introduction to Programming" course involving 286 freshmen at ETH Zurich. This year we decided to complement the traditional residential course (taught in German, with exercise sessions both in German and in English) with an online counterpart, taught in English and organized like a SPOC. The acronym SPOC was used the first time by Fox [5], who advocates the use of the MOOC "idea" to supplement classroom teaching as opposed to replacing it. In addition to the traditional frontal (live) lectures, exercise sessions, and home assignments, the SPOC provided an online, streamlined version of the above: lectures divided in relatively short lecture segments (17 minutes on average), quizzes both inside and outside the lecture segments, and programming exercises providing automatic feedback. We also introduced badges (awarded for submitting 100% correct answers in quizzes) to increase students' motivation. We summarize the results as follows:

- Students liked the course and responded with enthusiasm to the SPOC;
- Attendance to the live lectures remained stable with respect to the previous year;
- Attendance to each online lecture segment involved 71% of the total number of students on average;
- The average number of attempts to solve the online quizzes was almost five times bigger than the average number of views per online lecture;
- For any given quiz, an average of 48% of the first semester students enrolled in the online course scored 100%, and got the corresponding badge;
- On average, students worked on half of the given programming exercises, and 22% of them worked on 75% to 100% of all the exercises.
- In most cases, students completed programming exercises in a single session lasting an average of 20 minutes, and in 80% of the sessions students' solutions passed all tests.

The above findings suggest that the ideas of awarding a completion badge in quizzes and providing immediate and detailed feedback in programming exercises is effective in motivating the students.

The paper is organized as follows: Section 2 provides some background on our residential "Introduction to Programming" course and its structure; Section 3 describes the software platform we used and the personalizations and additions we provided ad hoc; Section 4 summarizes the data we collected, and our related comments; Section 5 contains our final comments and lessons learned from the experience.

ITiCSE'14, June 21–25, 2014, Uppsala, Sweden.

http://dx.doi.org/10.1145/2591708.2591759.

Table 1: Students' initial computing experience

Computer experience	Number of Students	%
Less than 2 years	0	0
2-4 years	4	2
5-7 years	22	8
7-9 years	69	25
More than 10 years	178	65
Total	273	100

Table 2: Students' initial programming experience

Programming Experience	%
Never programmed	16
No O-O experience	25
Less than 100 classes	53
More than 100 classes	6
Total	100

2. OUR "INTRODUCTION TO PROGRAMMING" COURSE: STRUCTURE AND STUDENT POPULATION

The residential course is 14 weeks long, excluding exams, which happen approximately 8 months after the course ends. The course includes four hours of live lectures per week over two days, and two hours of exercise sessions per week. The live lectures are of two kinds: traditional, frontal lectures and more interactive, hands-on exercise sessions. There are 10 home assignments and 2 mock exams to simulate a realistic exam setting. The exercise groups typically include 20 students each and are differentiated by skill level and language (English or German). Differentiating the exercise groups according to students' self-assessed skill levels (beginners, intermediate, and advanced) has proven to be quite effective in the last five years, because it enforced more homogeneous groups, which in turn helped to keep students with different backgrounds interested and motivated. Without this differentiation, advanced students tended to loose interest when teaching assistants were slowing down to explain basic concepts to beginners, while beginners tended to become frustrated when teaching assistants were speeding up to keep advanced students interested. The preliminary self-assessment questionnaire was useful to know also about the students' backgrounds.

We now summarize the results of this year's questionnaire. Table 1 clearly indicates that all the students have been significantly exposed to computers (mainly for Internet browsing, social networking, and games) for an average of 8 years (we also know that all of them have either a computer at home or a laptop). Though hardly surprising, it is worth mentioning that we have been having similar results since 2004, so these numbers appear to be rather stable.

To tailor the course to the student population, it is important to know about previous programming experience. The data we gathered on students' such experience is in line with what we observed in previous years. Table 2 shows that more than half of the students attending the residential course come with some object-oriented programming experience, while only 16% have never programmed before. In addition, 18% of the students have a part-time job that requires some programming.

Table 3: Previous knowledge of programming languages

Programming language	%
Java	23
C#	13
PHP	12
JavaScript	10
C++	8
Eiffel	7
Python	7
Others	20
Total	100

Table 4: Students' initial self-assessment

Desired level experience	Number of Students	%
Beginner	99	36
Intermediate	126	46
Advanced	48	18
Total	273	100

Table 3 shows the aggregated data about relevant programming language knowledge (we only considered answers claiming a "good" and "very good" knowledge) regardless of the knowledge distribution among students. To put the information in the right context, 41% of the total number of students enrolled in the course declared such a (good or very good) knowledge, and among them, the average student declared to know 2.5 programming languages. From the data it is clear that we have to deal with a wide selection of programming languages. This reinforces our choice to focus on the Eiffel programming language [4, 12] that, although not the most adopted for "Introduction to Programming" courses, makes it easier to express and teach the object-oriented concepts and methodology, and allows a gentle introduction to program verification through the Design by Contract technique [10].

Finally, in Table 4 we see students' self-assessed skill level that we used to assign them to a corresponding exercise group in order to avoid the issues of too heterogeneous exercise groups explained earlier.

Our approach to teaching "Introduction to Programming" is an extension of the "Inverted Curriculum" approach [11, 16]. In fact, we integrate the exercises based on the main GUI application (present in the residential course's home assignments) with smaller, self-contained examples and case studies. These allow students to focus on the main concepts in isolation, without the noise that a bigger application inevitably brings. The main application remains useful because it mimics the complexity of software they can find in real world applications, and lets the students cope with the software development activity that has been known for a long time to be the most relevant cost-wise: software maintenance [8].

3. OUR CUSTOMIZED ONLINE LEARNING PLATFORM

The platform we used was made of two distinct parts: a Moodle [2] installation, enhanced with a plugin we programmed to provide quizzes integrated in the online lectures, and a service infrastructure for developing and executing Eiffel programs in the browser, both running on our servers.

3.1 Virtual Learning Environment

For implementing the SPOC we used the Moodle platform, because it is free, open source, widely used, extensible, actively maintained, and with a large user community.

The online course sequence is linear and consists of 14 lectures. Every online lecture is divided into one or more segments (of variable duration), one or more quizzes, and zero or more programming exercises. Topics' durations vary between 5 and 40 minutes, with an average of 17 minutes. To tackle the well known issues deriving from reduced attention span (see for example [13] for an old but still valid reference), we embedded quizzes in the longer online lecture segments. The quizzes provide immediate feedback, can be attempted an unlimited number of times and are not graded. Their purpose is twofold: to allow attention span recovery by breaking the online lecture flow, and to test short-term topic comprehension.

The quizzes for the online lectures appear in a dedicated area next to the (paused) video presentation, allowing for a side-by-side visualization of the quiz and the relevant lecture material. The quizzes are standard multiple choice quizzes, consisting on average of 5 questions with each question offering 7 possible answers.

In general, we designed quizzes and exercises to be useful both in the short term (after taking an online lecture) and in the long term (for reviewing material for the exam, that in our case happens 8 months after the course end).

It is noteworthy to compare the differences between our approach and the "closed laboratory" as advocated by the ACM/IEEE Joint Curriculum Task Force [3, 19]. A closed laboratory model is defined as a:

> "scheduled, structured, and supervised assignment that involves the use of computing hardware, software or instrumentation for its completion. Students attend a closed lab by attending a scheduled session, usually 2-3 hours long, at a specific facility. Supervision is provided by the instructor. [...] Closed laboratories are particularly important in situations where the assignment relies on instructor-student interaction or a team effort among students to complete the work."

In our case the online exercises were intended as mostly optional self-study, they certainly did not rely particularly on instructor-student interaction (though some interactions actually took place, mostly by email) and they typically did not require a team effort among students.

3.2 In-browser programming exercises

We used programming exercises as another form of student feedback for some of the more advanced online lectures. The exercises were designed to train the specific topics, e.g. recursion, by providing an incomplete program that needed to be completed by the students. The correctness of students' solutions was evaluated automatically by running a test-suite on their submitted programs. The number of passing and failing test cases, together with the individual test inputs and outputs, were reported back to the students and they could resubmit new solutions, as many times as they liked. We persisted each student's best solution to allow them to review their submission at a later time. Finally, the programming exercises provided references to the relevant documentation of library classes and APIs for each programming exercise.

The infrastructure used for the programming exercises is a self-contained web service that we developed for this SPOC. The service, publicly available [1], provides basic functionalities for writing and running programs in the browser, thus allowing students to learn and train programming skills without the need to install any software.

For each programming exercise, a lecturer uploads a template program code together with a test suite. Each exercise is then accessible on a separate web page that features a syntax-highlighted code editor, options for compiling and running the program, as well as an output window for displaying the results of running the test suite. And example of such a page is shown in Figure 1. Finally, the exercise pages are integrated into the main SPOC pages, similar to the YouTube lecture videos, through HTML frame tags.

3.3 Assessment of student perception of the SPOC

Overall, students liked the course. With respect to the SPOC section, from Table 5 we see that for most of the students who answered the five questionnaires (in total we collected 318 answers for the various online lectures, quizzes, and programming exercises), the online lectures and quizzes were substantially just at the right difficulty level. The programming exercises were assessed as "too hard" by 23% of the respondents. This is consistent with our experience from previous years.

3.4 Participation

Our SPOC was de-facto open for enrollment to everybody. However, we did not advertise it, because we wanted to focus on our students during the first attempt of running this online course. A the end of the online course we had 327 enrollments, of which 286 were first semester students (determined through analysis of ETH Zurich email domains). Thus, apart from a few early testers and some outsiders constituting 12% of the enrolled participants, the majority of participants of the online course were ETH students.

Table 6 reports the hits per online lecture segment. Assuming that each student accessed each segment once or not at all, these data suggests that each segment was, on average, accessed by 71% of the students. Considering that "Introduction to Programming" students at our university take the exam approximately eight months after having attended the residential course, some may not have done all the exercises or attended all the online lectures during the course period, which is the period we used to collected the data presented here. This implies that our numbers are likely to be conservative estimates.

Table 6: Hits for online lectures, quizzes, and badges awarded. Column "Hits" reports how often a unit was viewed; column "No. segments" reports how many segments were part of a unit; "Quiz hits" reports how quizzes were viewed; "No. quizzes" reports how many quizzes were part of each unit; "No. badges awarded" reports how many badges were awarded in each unit. The second part of the table shows average values for *hits per unit, *hits per segment***, *hits per quiz*†, and *badges per quiz*‡.**

Unit	Hits	No. segments	Quiz hits	No. quizzes	No. badges awarded
Unit 2	968	1	1597	1	254
Unit 3	588	1	2374	1	191
Unit 4	866	2	1790	1	156
Unit 5	915	3	1406	1	169
Unit 6	859	3	1337	1	139
Unit 7	585	3	703	3	488
Unit 8	1191	5	1255	2	314
Unit 9	654	3	935	1	142
Unit 10	870	5	1114	5	601
Unit 11	213	1	803	1	123
Unit 12	228	2	550	1	77
Unit 13	187	1	661	1	131
Unit 14	636	3	847	1	116
Unit 15	434	5	548	1	67
Unit 16	542	4	733	1	65
Total	9736	42	23583	22	3033
Average	649*	232**	—	1072†	138‡

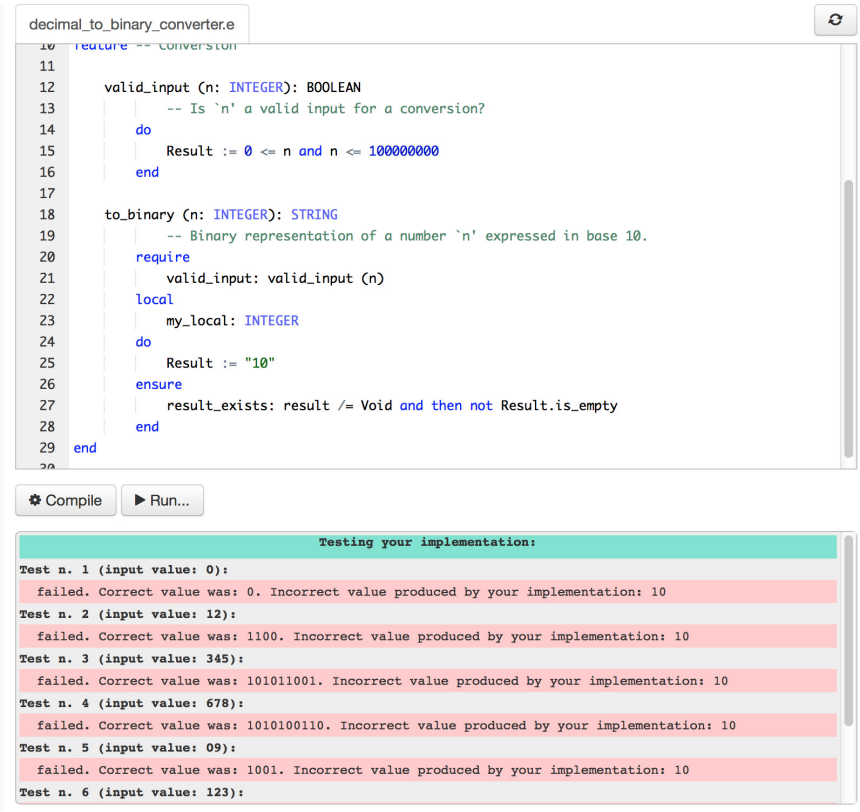

Figure 1: Screenshot of an in-browser programming exercise, showing the code editor and the output window which displays the passing and failing test cases.

Table 5: Student evaluation for SPOC online lectures, quizzes, and programming exercises (%)

Assessment	Lectures	Quizzes	Progr. exercises
Too easy	11	11	9
Just right	84	84	69
Too hard	5	5	23

4. COLLECTED DATA AND RESULTS

4.1 Assessment of quizzes

While we allowed just one submission for the standard homework assignments that were independent of the SPOC, we allowed unlimited attempts for the online quizzes, to focus students on the formative aspect of the assessment [15, 17]. With the same objective in mind, we provided detailed feedback in most of the quiz answers [6]. Interestingly enough, we received feedback from the students who asked for even more feedback to foster their learning. Every quiz was made of several questions, and for each quiz attempt the questions were presented in different order and the corresponding answers randomly shuffled. Each quiz attempt was independent from the previous attempts, so students had to provide previously given correct answers again. We intended the quizzes to be not too time consuming (5 to 15 minutes each) and intended to help consolidate the corresponding online lecture topics. We also added to the quizzes a gamification aspect [7]: to motivate students to solve each quiz 100% correctly—a goal we decided to be reasonable given the quiz complexity—we designed and awarded an electronic badge for each entirely correctly solved quiz. There were no penalties for multiple quiz attempts, to encourage motivated students to get the associated badge if they really wanted to.

Indeed the quiz data in Table 6 show that students on average attempted quizzes almost five times more than they viewed online lectures. This is a surprising result, suggesting that the choice of awarding such badges for correct completion may have worked out well. Another interpretation could be that students might have clicked randomly on the quiz answers until they got the completion badge, without actually learning anything. Though this is in principle possible, we believe it is very unlikely: the probability for such students of achieving a 100% score on our quizzes (a quiz has, on average, 5 questions with 7 randomly shuffled answer options per question) is only 0.006%. Thus, it is clear that it would take many more than five attempts to achieve a perfect score in the quiz.

An interesting aspect emerging from our data is that students of our online course took many of the quizzes despite of the fact that quizzes did not influence the final grade. This suggest that the students were not only trying to optimize their learning efforts towards achieving good grades [14, 18] but were also motivated to prove to themselves that they can solve the various quizzes and receive all the badges.

Table 6 shows that for each quiz, an average of 138 students scored 100% and got the corresponding badge. These 138 students constitute 48% of the total number of first semester students enrolled in the online course. An interesting result, considering that only 20 students (6%) participated in all the online course activities.

While the quizzes fall under the category of multiple choice, they vary considerably in structure. For example, to test understanding of source code, we have one type of quiz in which students have to fill in the blanks in an existing incomplete program by choosing among options in several drop-down menus (one option each menu). The catch is that given a set of drop-down menu options, many of them can be correct if considered in isolation. However, only one of them is correct when considered in combination with the options in the other menus.

4.2 Assessment of programming exercises

We also collected basic usage statistics for the programming exercises. For the analysis shown in this section, we focus on the main SPOC target group, consisting of the 286 first semester students. Out of those 286 students, a total of 105 (37%) worked on at least one of the four programming exercises we provided.

For all numbers reported in this section, we removed programming sessions that lasted less than two minutes as they are likely to represent students "having a quick look" but not actually working on the exercise.

We found that, on average, students worked on 1.7 out of the 4 programming exercises. About 22% completed three or four of the exercises.

The work on the exercises spreads over 190 distinct sessions, implying that, on average, a student worked in 1.8 distinct sessions (where a session is recorded as a browser session). This number indicates that, in most cases, students complete an exercise within a single session and do not return at a later time to try the programming exercise again. The average length of a student session, starting from the first compilation until the last compilation/execution, was about 20 minutes with at least 50% of all sessions lasting longer then 15 minutes.

Looking at the number of compilations (2004) and executions (826) of programs that were written by students, we observe, on average, 2.5 times more compilations than executions. We interpret this as an indicator that it takes students 2-3 attempts before they write syntactically correct code that can then be executed. This is in line with the amount of effort we expected from the average student to solve these kinds of programming exercises.

We believe that the feedback about passing and failing test cases motivates students in similar way the badges motivate them to complete the quizzes. We found that in almost 80% of all cases, students continued to work on their exercise solution until it passed all test cases.

While not all students worked on the programming exercises, individual feedback was mostly positive. For example, a student reported that

> "The cool thing about MOOC is you don't need [an IDE] or even a Computer but ... you can't debug it unless you copy it into [the IDE]."

This quote is in line with our impression that the convenience of in-browser programming exercises is appreciated but more advanced features should be added to improve the student's experience.

5. CONCLUSIONS

Research shows that even though MOOCs offer autonomy and connectedness, the lack of support and guidance can be

an issue [9]. With our Autumn 2013 "Introduction to Programming" course, we tried to take the best of residential courses and MOOCs by integrating a SPOC into our standard residential course. The results have been encouraging: students' response was positive, and so were the participation and completion rates, given that the material was not mandatory. In addition, we are left with the impression that gamification seems to motivate students to learn.

Individual students' comments included:

> "The MOOC was helpful to understand the theory. "
>
> "The MOOC is an awesome thing. However, some things could still be improved. Shorter videos and more precise titles are desirable. This way, it could be used as a better learning aid, i.e. looking up a specific topics, watching the same section multiple times."
>
> "The MOOC is a very good idea to repeat/test what you have learned."
>
> "The MOOC, which was introduced as a secondary learning instance, complements the lecture perfectly and should absolutely be continued and advanced in the next year. "

As future research, we will analyze the final exam results (once the students have taken the exam) to measure the effect of the SPOC on the students' performance. We are also planning to advertise the online course as a MOOC to the outside world, so others can attend it or use it for teaching. Finally, we will use the insights and findings from this paper to improve our online course for its next iterations in the future.

6. ACKNOWLEDGMENTS

The Authors would like to thank Andre Macejko for his work on the implementation of the Moodle plugin for quizzes in online lectures, Carlo A. Furia for his comments on the paper draft, and the anonymous referees for their useful remarks.

7. REFERENCES

[1] Eiffel 4 mooc. `http://se.inf.ethz.ch/data/spoc/`. Accessed April 14, 2014.

[2] Moodle virtual learning environment. `https://moodle.org`. Accessed April 14, 2014.

[3] C. Chang. Computing curricula 2001 computer science: The joint task force on computing curricula. http://www.acm.org/sigcse/cc2001/cc2001.pdf. Accessed April 14, 2014.

[4] E. Committee. *ECMA International Standard 367: Eiffel Analysis, Design and Programming Language*. ECMA International, 2006.

[5] A. Fox. From MOOCs to SPOCs. *Communications of the ACM*, 56(12):38–40, Dec. 2013.

[6] G. Gibbs and C. Simpson. Conditions under which assessment supports students' learning. In *Learning and Teaching in Higher Education*, 2005.

[7] J. J. Lee and J. Hammer. Gamification in education: What, how, why bother? *Academic Exchange Quarterly*, 15(2), 2011.

[8] B. P. Lientz and E. B. Swanson. *Software Maintenance Management*. Addison Wesley, Reading, MA, 1980.

[9] J. Mackness, S. F. J. Mak, and R. Williams. The ideals and reality of participating in a mooc. In *7th International Conference on Networked Learning*, pages 266–274, 2010.

[10] B. Meyer. Applying design by contract. *Computer*, 25(10):40–51, 1992.

[11] B. Meyer. Towards an object-oriented curriculum. In *TOOLS (11)*, pages 585–594, 1993.

[12] B. Meyer. *Object-Oriented Software Construction, 2nd edition*. Prentice Hall, 1997.

[13] J. Middendorf and A. Kalish. The "change-up" in lectures. *The National Teaching & Learning Forum*, 5(2), 1996.

[14] C. Miller and M. Parlett. Up to the mark: a study of the examination game. *Guildford: Society for Research into Higher Education*, 1974.

[15] P. F. Mitros, K. K. Affidi, G. J. Sussman, C. J. Terman, J. K. White, L. Fischer, and A. Agarwal. Teaching electronic circuits online: Lessons from MITx's 6.002x on edX. In *ISCAS*, pages 2763–2766, 2013.

[16] M. Pedroni and B. Meyer. The inverted curriculum in practice. *SIGCSE Bull.*, 38(1):481–485, Mar. 2006.

[17] V. J. Shute. Focus on formative feedback. *Review of Educational Research*, pages 153–189, Mar. 2008.

[18] B. Snyder. The hidden curriculum. In *Learning and Teaching in Higher Education*, 2005.

[19] A. B. Tucker. *Computing Curricula 1991: Report on the ACM/IEEE-CS Joint Curriculum Task Force*. ACM Press, 1991.