MOOC as Semester-long Entrance Exam

Arto Vihavainen, Matti Luukkainen, Jaakko Kurhila
University of Helsinki
Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
Fi-00014 University of Helsinki
{ avihavai, mluukkai, kurhila }@cs.helsinki.fi

Originally appeared as: Arto Vihavainen, Matti Luukkainen, and Jaakko Kurhila (2013): MOOC as semester-long entrance exam. In Proceedings of the 13th annual ACM SIGITE conference on Information technology education (SIGITE '13). ACM, New York, NY, USA, 177-182.

Abstract

MOOCs (massive open online courses) became a hugely popular topic in both academic and non-academic discussions in 2012. Many of the offered MOOCs are somewhat "watered-down versions" of the actual courses given by the MOOC professors at their home universities. At the University of Helsinki, Department of Computer Science, our MOOC on introductory programming is exactly the same course as our first programming course on campus. Our MOOC uses the Extreme Apprenticeship (XA) model for programming education, thus ensuring that students are proceeding step-by-step in the desired direction. As an additional twist, we have used our MOOC as an entrance exam to studies in University of Helsinki. In this paper, we compare the student achievement after one year of studies between two cohorts: the MOOC intake (n=38) and the intake that started their studies during the fall (n=68). The results indicate that student achievement is at least as good on the MOOC intake when compared to the normal intake. An additional benefit is that the students admitted via MOOC are less likely to drop out from their studies during their first year.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and Information Science Education *Computer Science Education*

General Terms
Experimentation

Keywords
entrance exam, admission, first-year experience,
student achievement

1 Introduction

MOOCs or massive open online courses have been a source for an intense debate recently in academia, both in administration and among teachers (see e.g. [5]). MOOCs come in a variety of forms; however, most of the current high-profile MOOCs tend to be based on short lectures (8-12 min videos, animations and screencasts) interspersed with quizzes that are used to keep up the students' attention to the learning material[1]. A key issue in MOOCs is to facilitate and allow massive attendance.

MOOCs have been aptly described as "textbooks on steroids" [6]. In other words, the students that are successful in MOOCs tend to be autodidacts, to the extent that e.g. more than 70% of the starting MOOC students already have an undergraduate or postgraduate degree [13].

Our MOOC at the University of Helsinki Department of Computer Science differs from typical MOOCs in two key aspects [16]:

- Students start by installing a real-world programming environment and start to program immediately. All learning materials are built to support hands-on programming. The emphasis is heavily on a learning process that allows and requires the learners to produce working solutions. There are hundreds of programming assignments that the students are expected to construct during the course.

- By successfully completing the MOOC and participating in an interview, a student is granted admission to the university to major in Computer Science.

In Finland, the students choose their major before entering a university, making the decision often based on a relatively vague idea of the area and whether the studies suit the student. Using a traditional entrance exam as a way to select students provides insufficient results[2] as some of our first-year students fail to succeed in e.g. the very first required programming courses, effectively forcing them to seek another study.

The most important part of MOOC as semester-long entrance exam is the *process of learning to program*, during which the student sees if CS/IT is the desired area of study for her. Successfully completing the MOOC provides us and the student herself the evidence that shows that she has the aptitude for CS/IT.

In practice, our MOOC is exactly the same course as the entry-level programming course in our university. This in itself acts as a validation measure to see whether a student is able to handle the first and often the most challenging courses.

[1] A notable exception are so-called connectivist MOOCs that rely more on facilitated discussions among networked learners [14, 7, 12].

[2] Attempts to pinpoint identifiable markers for aptitude to succeed in CS/IT studies have yielded non-conclusive results (see e.g. [10, 1, 15, 8]), making it impossible to derive a set of markers for revising the traditional entrance exam.

In the first 18 months of operation, our MOOC has had 2109 participants, from which some 200 students have applied for a study position. The MOOC in programming was first used as an entrance exam in spring 2012. In addition to the new admission path via MOOC, the traditional admission procedure was kept intact and also offered to high-school students.

In this paper, we compare the success of students admitted via a MOOC to the traditional entrance exam-based intake. As our work on using MOOCs as an entrance exam to university studies has been active only for a short while and we are still adjusting the level that we require from the MOOC participants for them being admitted, we have deliberately chosen not to include statistical analysis to avoid drawing premature conclusions at this stage.

2 Educational system in Finland

Before starting undergraduate higher education in Finland, students typically have 12 years of schooling. During those 12 years, there is only one standardized test: the matriculation exam after the 12th grade. Major subject is chosen before starting the University studies.

Universities can use the results of the matriculation exam to grant study rights. However, most study disciplines in the universities use the matriculation exam results only as a small addition to university- and subject-specific entrance exams, especially in highly desirable subjects. In STEM subjects, the admission is typically more generous due to the lack of applicants. Admission can be granted based on either 1) solely the entrance exam, 2) solely the matriculation exam, or 3) a combination score from entrance exam and matriculation exam. At the Univ. Helsinki Dept of Computer Science most of the admitted students have taken the entrance exam (402 applicants in 2013) and received some extra points based on their matriculation exam. Entrance exams for CS/IT – and other subjects as well – are classic pen-and-paper tests conducted in a lecture hall under strict surveillance so that candidates are using only their brains to answer the exam questions.

As computer science (computing, or any IT-related topic) is not among the mandatory study subjects in high school in Finland, it is not part of the matriculation exam [3]. Therefore, the entrance exam to CS/IT does not contain programming per se; instead it contains logical problems and essay writing. Many of the students who are admitted to computer science do not have an accurate image of the subject, and many drop out soon after their studies have started.

Another issue worth noting is that there are no tuition fees for anyone in Finland (from elementary schools to universities). Instead, the government supports students by a monthly allowance for living expenses, including rent

[3] Many schools offer computing as an elective course. However, as there is no national curriculum for courses in computing, courses often concentrate on the use of computer applications and computer literacy. The situation in Finland as such resembles many other countries, e.g. USA [20].

support. As CS/IT is not among the most highly sought-out study subject, some students apply for CS/IT as a fallback position, and accept the study right in order to get the student benefits. Instead of studying CS/IT, they use the extra year for e.g. preparing for an entrance exams to a more preferable study subject.

In order to alleviate the problem of having an incorrect mental image of CS/IT studies, we wanted to allow high-school students (esp. grades 10 to 12) in Finland to experience CS/IT studies. Therefore, we opened up our introductory programming course (CS1) to the whole country[4], targeting especially high-school students who have no programming education in their schools, or who seek more advanced courses than their local high school offers[5].

The most significant benefit is that the MOOC participants get a more realistic view of the studies they would be encountering if they took CS/IT as a major subject, and can themselves evaluate if they are up to it. By completing the MOOC in programming, the students show us at the department that they are both competent and persevering enough to study CS/IT. Therefore, it is only natural to grant those students full study rights for a degree.

3 MOOC as an Entrance Exam

Starting the MOOC is straightforward, as there is no need to provide any other information than a valid email address when registering. If the student seeks admission, she is required to enter full personal information. The option for applying for the study right is available for the first two months.

3.1 Course Content and Pedagogy

The MOOC in programming is content-wise exactly the same as our CS1, which is taught in Java using an objects-early approach. The course contains 12 weekly exercise sets, and covers topics typical to any introductory programming course; assignment, expressions, terminal input and output, basic control structures, classes, objects, methods, arrays and strings, advanced object oriented features such as inheritance, interfaces and polymorphism, and familiarizes students with the most essential features of Java API, exceptions and file I/O[6].

During the MOOC, the participants work on over 150 programming exercises, which are further split into over 350 tasks. The students that are applying for the study rights must correctly solve 80% or more of the weekly tasks in order to be invited to the interview. The material is handed out online in a book-like

[4] As is typical for MOOCs, there are no restrictions for participation. Our MOOC is in Finnish language, so the natural audience is mostly in Finland.

[5] In case a participant does not want to apply for a study right but would like to have a certificate of accomplishment, we have facilitated the schools in Finland to provide examinations. High schools use the certificate for granting school credits.

[6] The course material and exercises are available at http://mooc.fi and licensed under the Creative Commons BY-NC-SA -license.

format, with a few screencasts, and its sole purpose is to help the students work on the exercises; the main working method for the students is programming.

The learning-by-doing orientation comes from using the Extreme Apprenticeship (XA) [17] method in the course implementation. XA is based on cognitive apprenticeship [3, 2] and approaches programming as a craft that needs to be honed continuously. Two core values in XA are "practice as long as necessary" and "continuous feedback". In our earlier XA-based courses [11], the feedback has been provided by human advisors (teachers). In the MOOC, the participants program in an industry-standard programming environment that contains a plugin, which provides help for the students (for additional details, see [18]).

3.2 Interview and Programming Task

Once the students have worked through the required number of programming tasks, they are invited to an interview. The interview is a two-part process: first, the students work on a programming task in a live setting, and after that are interviewed by two members of the faculty. The programming task is done in a lab, where a supervisor can help participants with e.g. operating system or programming environment-related issues, and correct potential misunderstandings regarding the task. The students are free to use any available material which can be found on the internet, e.g. the course material. However, asking for help in solving the programming task is not allowed.

The participants had a total of 2 hours for the task, which was as follows for the interview held during spring 2012.

Programming task: a text analyzer

Create an application that can be used to analyze text file contents. The application should contain at least the following features:

- calculating the number of words in a file
- finding and printing the most common word(s) in a file
- finding and printing the longest word in a file

If you wish, you can also create additional features.

The program should be able to analyze several files during a single execution, and it should also be able to handle large files. You can test your application for example with Kalevala, which is available at
http://www.gutenberg.org/cache/epub/7000/pg7000.txt

You can decide what sort of a user interface the application provides, however, we suggest that you build a text-based user interface. Below is an example of how the application could work:

```
Enter filename, empty input exits the program
> kalevala.txt
commands: longest, words, most-common, help
command > words
67443
command > longest
longest word is: kautokengän-kannoillansa
command > most-common
most common word is: on
>
finished processing kalevala.txt

Enter filename, empty input exits the program
> test.txt
commands: longest, words, most-common, help
command > help
commands: longest, words, most-common, help
command > words
7
command >
finished processing test.txt

Enter filename, empty input exits the program
>
Thank you!
```

After the programming task, the students are interviewed for up to 30 minutes by two faculty members. The faculty members discuss the students' program design choices and possible issues with e.g. performance with the student. During the interview, the faculty also attempts to form an understanding of the student's background, and reasons for applying to the department of computer science. Things that are of interest are e.g. existing programming background, the student's vision regarding her life after five years from now, and existing educational background.

3.3 Selection of Students

During spring 2012, most of the students that did over 80% of the exercises in the MOOC in programming and applied for study rights also fared well in the actual interview. Most of the participants were able to complete the programming task fully, and only a handful of the participants had issues with e.g. program design

or did not have a working program at all. Out of the 52 students that applied for a study position during spring 2012, 49 study rights were granted. Out of the 49, 38 students started their studies during fall 2012, and the remaining 11 had varying reasons not to start their studies: they are still in high school, they postponed the start due to the mandatory military service, or they took another, preferred study position.

The number of applicants via the traditional path has been in hundreds for years. Therefore, we are not expecting an uncontrollable need to scale up the interview process. Currently, the interviews involved with the MOOC entrance have been conducted by two faculty members without extra resources.

4 Data

Our data contains study records from students that have started their studies at the Department of Computer Science at the University of Helsinki in August 2012. As some of the students postpone their start due to the military service, focus on other studies than Computer Science, or have transferred courses from earlier studies (e.g. open university), we include only students that have either attempted or completed the introductory programming course during the academic year 2012-2013.

The study records cover the period from August 1, 2012 to May 24, 2013. We examine two separate groups. The first group (MOOC, n=38) contains students that have been admitted via the programming MOOC that was organized during spring 2012. The second group (NORM, n=68) contains students that were admitted via the traditional path, namely the entrance exam, matriculation exam, or a combination of both. The MOOC group has the introductory programming and advanced programming courses (a total of 9 ECTS[7]) included in the data, as the courses have been added to students' records when they were granted study rights, i.e. 1st of August 2012. We offered the MOOC for all students that were admitted as well. The second group (NORM) does not include students, who took the MOOC during the summer (n=15), as their effective study time would be 3 months longer than the other students in NORM group, causing additional deviation in the data.

For each study subject (CS/IT, Math, all), we report the number of credits, number of courses passed, number of courses failed, and grades for each category. The grades range from 1 (pass) to 5 (excellent), and the grade averages exclude failed courses. Our university does not force courses to be graded on a bell curve. On the contrary, student grades are based on the true performance of the student using an explicit criteria.

When looking at the data, one should keep in mind that the study path for first-year students is designed for students taking the programming courses during the first semester. This means that the students that have been admitted via MOOC have received no benefits from a tailored study path.

[7]European Credit Transfer and Accumulation System. An academic year corresponds to 60 ECTS, and one ECTS credit point equals 25-30 hours of student work.

	CS/IT Courses	
	MOOC (n=38)	NORM (n=68)
Credits		
overall	1257	1629
mean	33.08	23.96
std	11.32	15.3
median	32.5	24
Courses passed		
overall	346	452
mean	9.11	6.65
std	2.95	4.02
median	9	7
Courses failed		
overall	66	157
mean	1.74	2.31
std	1.83	2.37
median	1	2
Grade stats		
mean	4.04	3.79
std	1.15	1.2
median	4	4

Table 1: Student performance in CS/IT-related courses.

4.1 CS/IT Courses

Table 1 contains the students' performance in CS/IT courses. When considering the number of credits that students have gathered during the study period, the average is almost identical when we include the knowledge that MOOC students have taken the introductory programming courses earlier. The standard deviation in the number of credits, which is higher for the NORM group, indicates that there is more variance within the NORM group. In essence, it indicates that there are students that end up failing their first programming courses and do not proceed at all, as well as students, who fare well in their studies.

On average, the students admitted via a MOOC pass more CS/IT courses than the NORM group, and end up failing less courses. On average, MOOC students have one fail per five passed courses, while the NORM group has one fail per three passed courses. The standard deviation in both passed and failed courses is also smaller for the MOOC group; on average, the MOOC students fare better than the NORM students. This is also seen in the grade statistics; although there is not much difference, and the median grade is 4 on a scale from 1 (pass) to 5 (best) for both groups, the average grade is slightly higher for the MOOC group.

	Math Courses	
	MOOC (n=38)	NORM (n=68)
Credits		
overall	273	350
mean	7.18	5.15
std	7.49	7.63
median	5	0
Courses passed		
overall	46	62
mean	1.12	0.91
std	1.19	1.25
median	1	0
Courses failed		
overall	30	50
mean	0.79	0.74
std	0.74	0.66
median	1	1
Grade stats		
mean	3.39	3.37
std	1.2	1.45
median	4	4

Table 2: Student performance in mathematics courses.

4.2 Mathematics Courses

In Table 2, we see the students' performance in mathematics courses. Although mathematics is not a mandatory minor subject, completing at least 10 ECTS of mathematics is mandatory. Typically, students enroll in a course called Introduction to University Mathematics, which covers the essential mathematics required for the course on Data Structures (CS2), where e.g. algorithm run-time analysis is one of the focus areas.

On average, both student groups have completed at least 5 ECTS of mathematics during their first year of studies. The MOOC students have taken over 7 ECTS worth of mathematics, while NORM students have 5.15 ECTS. Note, however, that the standard deviation is high for both groups, which means that it is very likely that there are students in both groups that have either not passed any mathematics courses, or have passed more than one mathematics course.

When looking at the number of passed courses, the median for the MOOC students is 1, and the median for NORM students is 0. This means that one half or more of the NORM students have not succeeded in passing any mathematics courses. This is problematic, as although mathematics is not a formal requirement for CS2, it is highly beneficial for students to understand the contents of

	All Courses	
	MOOC (n=38)	NORM (n=68)
Credits		
overall	1675	2296
mean	44.08	33.76
std	17.58	21.96
median	43	32.5
Courses passed		
overall	434	599
mean	11.42	8.81
std	4.24	5.27
median	11	9
Courses failed		
overall	101	214
mean	2.66	3.15
std	2.16	2.73
median	2	3
Grade stats		
mean	3.94	3.73
std	1.13	1.18
median	4	4

Table 3: Student performance in all courses.

the Introduction to Mathematics course as they take on Data Structures.

The grade averages for both groups are almost alike; the only difference being the slightly higher standard deviation for the NORM group.

4.3 All Courses

Table 3 contains information on all the courses that the students have taken during their first year of studies. It contains both the CS/IT courses and the mathematics courses, and in addition other courses that the students may have taken. Students are able to choose almost any course from any discipline, so minor studies vary a lot among the students. Among the students that have started their studies in 2012, we have students taking courses related to e.g. politics, economics, literature, psychology, languages and law.

Overall, the students in the MOOC group fare slightly better on average than the NORM group, but the NORM group has more variation. On average, the MOOC students have gathered 44.08 ECTS during their first year (35.08 if programming courses are not included), while the NORM students have gathered 33.76 ECTS. There is a small, but noticeable difference, and the median is 43 for MOOC (34 if programming courses are not included), and 32.5 for

NORM[8].

When looking at the number of courses passed, and the number of courses failed, the MOOC students fare better on average, while the NORM students have a larger variation. The MOOC students have one fail per four passed courses, while the NORM students have one fail for slightly less than three courses. Again, some students perform well, while others perform poorly. The grade statistics are almost alike, on average the grade of MOOC students is 3.91, while the grade average for NORM students is 3.71.

In addition, when considering the amount of students that have received less than 10 ECTS during their first two semesters, i.e. have done only the programming course or less, only one out of the 38 MOOC students did not complete anything outside the programming courses. When considering the students in the NORM group, a total of 12 students (17.6%) have gathered less than 10 ECTS. We must note that we consider only the students that started their studies and participated in the introductory programming course; in reality, the number is higher.

5 Discussion and Future Work

Our initial analysis of students that have been admitted via the MOOC indicates that they are failing less courses and gaining slightly more credits than the students admitted via the traditional path. However, lots of variance in the student groups exist, and both of the groups have so-called high performers and low performers. As we compared the MOOC students to students that have attempted or succeeded in the introductory programming courses during the academic year 2012-2013, our initial analysis excluded the admitted students that did not study at all (e.g. entered military service or started to study another subject at the university) or chose to start their studies early by participating in a voluntary MOOC during summer 2012.

At the University of Helsinki, Department of Computer Science, we receive some 500-600 study applications per year. A majority of the applicants seek a study right via the entrance exam, while some apply directly using their matriculation exam score. Typically less than 200 students are admitted, and of these, on average, less than 130 students accept the study right. Thus, CS/IT is not the number one choice for the study for many of the applicants. Moreover, some 20-30 students do not start any CS/IT courses, even if they accept the study right. When we compare these traditional figures to our first MOOC intake, in which over 93% of the applicants were accepted and started their studies accordingly, the MOOC intake is far superior in matching the students to an appropriate and desired area of study.

[8]It should be noted that the student should complete 60 ECTS per academic year in order to graduate according to the model curriculum. In practise, a slow start and advancement of CS/IT studies (as well as large dropout rate) is a common problem in Finland. Even the most competent students tend to start working in the IT industry while studying, thus delaying their graduation.

Having the students successfully perform introductory programming courses already before they start their studies gives the students a head start over their fellow students. It also acts as a preliminary verification on the students' motivation to study CS/IT. In addition, the students are not getting stuck to the "filter" of learning to program that is a cause for challenges for many in their early studies.

As the awareness of our MOOC as an entrance exam is increasing, we are currently in the process of increasing the number of students admitted via the MOOC. In spring 2013, a total of 66 students were admitted. In addition to improving the intake, we are also working on the students' first year experiences so that the MOOC students have more relevant courses to work on. Even though our MOOC has proven to be beneficial for us, we are not aiming to stack up on online education: we want all of our students to participate in the academic community and therefore emphasize the social support during the degree studies, helping them in the transition from a high school to the university [19].

We see a strong indication that one of the important success factors in first-year CS/IT studies is foundational programming skills. These skills can be practised already before the formal start of the degree studies. Universities with a similar admission system to ours that are facing challenges with student intake and performance (e.g. students dropping out during first semester, students not opting for CS/IT-studies) may benefit from a long-term programming exam, which is administered already during the high-school studies (cf. e.g. [4, 9]).

Acknowledgements

This research is partially funded by the Technological Industries of Finland Centennial Foundation. We gratefully acknowledge the anonymous reviewers for their valuable feedback.

References

[1] M. E. Caspersen, K. D. Larsen, and J. Bennedsen. Mental models and programming aptitude. In *ACM SIGCSE Bulletin*, volume 39, pages 206–210. ACM, 2007.

[2] A. Collins, J. Brown, and A. Holum. Cognitive apprenticeship: Making thinking visible. *American Educator*, 15(3):6–46, 1991.

[3] A. Collins, J. Brown, and S. Newman. Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In *Knowing learning and instruction Essays in honor of Robert Glaser*, volume Knowing, l of *Psychology of Education and Instruction Series*, pages 453–494. Lawrence Erlbaum Associates, 1989.

[4] T. Crick and S. Sentance. Computing at school: stimulating computing education in the UK. In *Proceedings of the 11th Koli Calling International*

Conference on Computing Education Research, Koli Calling '11, pages 122–123, New York, NY, USA, 2011. ACM.

[5] J. Daniel. Making sense of MOOCs: Musings in a maze of myth, paradox and possibility. 2012. http://www.academicpartnerships.com/docs/default-document-library/moocs.pdf.

[6] K. Devlin. The future of textbook publishing is us, 2012. http://devlinsangle.blogspot.fi/2012/08/the-future-of-textbook-publishing-is-us.html.

[7] S. Downes. What is a connectivist MOOC. 2012. http://www.connectivistmoocs.org/what-is-a-connectivist-mooc/.

[8] G. E. Evans and M. G. Simkin. What best predicts computer proficiency? *Commun. ACM*, 32(11):1322–1327, Nov. 1989.

[9] B. Franke, J. Century, M. Lach, C. Wilson, M. Guzdial, G. Chapman, and O. Astrachan. Expanding access to k-12 computer science education: research on the landscape of computer science professional development. In *Proceeding of the 44th ACM technical symposium on Computer science education*, SIGCSE '13, pages 541–542, New York, NY, USA, 2013. ACM.

[10] P. Kinnunen, R. McCartney, L. Murphy, and L. Thomas. Through the eyes of instructors: a phenomenographic investigation of student success. In *Proceedings of the third international workshop on Computing education research*, ICER '07, pages 61–72, New York, NY, USA, 2007. ACM.

[11] J. Kurhila and A. Vihavainen. Management, structures and tools to scale up personal advising in large programming courses. In *Proceedings of the 2011 conference on Information technology education*, SIGITE '11, pages 3–8. ACM, 2011.

[12] A. McAuley, B. Stewart, G. Siemens, and D. Cormier. The MOOC model for digital practice. 2010. http://davecormier.com/edblog/wp-content/uploads/MOOC_Final.pdf.

[13] MOOCs@Edinburgh Group. MOOCs @ Edinburgh 2013: Report nr. 1, 2013. http://hdl.handle.net/1842/6683.

[14] G. Siemens. What is the theory that underpins our MOOCs? 2012. http://www.elearnspace.org/blog/2012/06/03/what-is-the-theory-that-underpins-our-moocs/.

[15] Simon, S. Fincher, A. Robins, B. Baker, I. Box, Q. Cutts, M. de Raadt, P. Haden, J. Hamer, M. Hamilton, R. Lister, M. Petre, K. Sutton, D. Tolhurst, and J. Tutty. Predictors of success in a first programming course. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ACE '06, pages 189–196, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[16] A. Vihavainen, M. Luukkainen, and J. Kurhila. Multi-faceted support for MOOC in programming. In *Proceedings of the 13th annual conference on Information technology education*, SIGITE '12, pages 171–176. ACM, 2012.

[17] A. Vihavainen, M. Paksula, and M. Luukkainen. Extreme apprenticeship method in teaching programming for beginners. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, SIGCSE '11, pages 93–98. ACM, 2011.

[18] A. Vihavainen, T. Vikberg, M. Luukkainen, and M. Pärtel. Scaffolding students' learning using Test My Code. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, ITiCSE '13, pages 117–122, New York, NY, USA, 2013. ACM.

[19] P. Wilcox, S. Winn, and M. Fyvie-Gauld. 'It was nothing to do with the university, it was just the people': the role of social support in the first-year experience of higher education. *Studies in higher education*, 30(6):707–722, 2005.

[20] C. Wilson, L. A. Sudol, C. Stephenson, and M. Stehlik. Running on empty: The failure to teach k-12 computer science in the digital age. Association for Computing Machinery. 2010.