Informatics in Education, 2023, Vol. 22, No. 4, 635–650
© 2023 *Vilnius University, ETH Zürich*
DOI: 10.15388/infedu.2023.27

To Schedule or Not to Schedule: The Effects of Course Structure on Programming MOOC Performance

Erkki KAILA[1,2], Kjell LEMSTRÖM[2]
[1]*University of Turku, Department of Computing*
[2]*University of Helsinki, Department of Computer Science*
e-mail: erkki.kaila@utu.fi, kjell.lemstrom@helsinki.fi

Received: November 2022

Abstract. Massive Open Online Courses (MOOCs) have become hugely popular recently. MOOCs can offer high-quality education for anyone interested and equalize the whole education field. Still, there are different methodologies for running MOOCs. Coming up with the most suitable methodology benefits both students and teachers. In this study, we have limited the methodological focus to observing scheduled and unscheduled instances of similar MOOC courses. While unscheduled MOOC courses can provide flexibility, they also require self-regulated learning strategies for students to succeed. To observe this, we compare the effectiveness of scheduled and unscheduled programming MOOC courses to find the most effective methodology. For this, we compare the pass rates and grade averages of five instances (two unscheduled and three scheduled) of Python and Java programming MOOCs. The results show that while the attendance numbers are higher in the unscheduled versions, in the scheduled instances the pass rate is significantly better, and students' progression is much swifter. It also seems that the higher proportion of university students enrolled in a MOOC course positively affects the retention rate. Moreover, the students in the recent unscheduled Python version seem to score significantly higher grades than in its scheduled counterpart. Based on our experiments, the scheduled and unscheduled versions complement each other. Hence, we suggest that, whenever feasible, the maximal benefits would be gained if both types of MOOCs are run simultaneously.

Keywords: MOOC, shceduled vs. unscheduled, programming, student performance.

1. Introduction

MOOCs (Massive Open Online Courses) are online courses that are targeted to a large number of students. They usually are also offered free of charge, although in some cases extra services such as a certificate of completion, personal feedback, or access to some premium content are acquirable by a fee. In a typical arrangement, a MOOC course involves online materials, video lectures, interactive assignments, and quizzes. MOOCs are often delivered by universities or other educational institutions: The courses are designed to provide access to high-quality education to anyone with Internet access, regardless of their location or study rights in a university. Thus, MOOCs can be virtually accessed from anywhere at any time, providing students a chance to learn at their very own pace.

MOOCs have sometimes faced criticism for their high dropout rates and for their lack of interaction and engagement compared to traditional classroom settings, see e.g. Bartolomé and Steffens (2015). However, they have also been praised for their potential to make high-quality education accessible to people around the world, particularly those who might not otherwise have access to it, see e.g. Sanchez-Gordon and Luján-Mora (2016). Nevertheless, MOOCs continue to evolve and develop, with new platforms, technologies, and approaches emerging all the time.

MOOCs are considered to have emerged from the Open Education Resources movement that consisted of freely accessible and openly licensed documents and media targeted for teaching, learning, and research purposes (Petkovska *et al.*, 2014). In 2008, Stephen Downes and George Siemens coined the term MOOC to refer to their course entitled Connectivism and Connectivity Knowledge (Kovanovic *et al.*, 2015). With the course, they wanted to exploit the possibility for interactions between participants in a learning environment that offers more possibilities than the traditional courses do. In their set-up, 25 students attended the campus instance, and 2,300 students the online instance. In 2011, Peter Norvig and Sebastien Thrun from Stanford University offered their MOOC on Introduction to Artificial Intelligence to an initial enrollment of over 160,000 students. Over 20,000 students completed the course (Rodriguez, 2012).

Recently, in many countries, the mission and goals of tertiary education providers have been extended to educate the general public by offering open elementary courses on the education provider’s key expertise areas and supporting the lifelong learning of society. As this can be effectively implemented by providing MOOCs, this has further influenced the already mushroomed number of them and their familiarity within society. Also, new goals for MOOC courses have been created. For instance, they have recently been used as an extra intake mechanism for university student applicants. For example, Vihavainen *et al.* (2013a) presented a semester-long programming course working as an entrance exam. More recently, Pirttinen *et al.* (2021) introduced preliminary results of a project where applicants who complete 60 credits a year from a vast pool of MOOC courses can enroll without a further entrance examination. They also argue that well-planned and implemented MOOCs serve as a very effective marketing mechanism to reach more and better applicants to the study program.

One important attribute of a MOOC is whether it is scheduled or unscheduled, that is, does the course have tight deadlines, or is it available at any time without (virtually) any deadlines for a student to progress in the course? At the University of Helsinki, we have a relatively long history with MOOCs. We launched our first MOOC in programming in 2012. Currently, we have well over 100 ECTS[1] credits available in our MOOC pool for anyone to complete[2].

[1] *European Credit Transfer and Accumulation System*

[2] The citizens of Finland can collect ECTS credits, others may have a certificate of their achievement.

In this paper, we study the influence of the scheduled/unscheduled attribute on student performance in 2020 and 2021 instances of our programming MOOCs. The period under review coincided with the point where we also changed the basic teaching language of our degree from Java to Python. In 1997, Java substituted Pascal in our curriculum due to its platform independence, strong object-oriented programming support, and extensive standard library. Then, almost 25 years later, it was decided that Java, in turn, needs to be replaced by a more modern programming language, this time by Python. We considered Python to overpower Java as the first programming language to be learned because it has a more gentle learning curve due to its simple and intuitive syntax, its provision of dynamic typing (no need to declare the data type of a variable before using it) and its faster feedback loop (quick to write, test and debug).

We studied the scheduled/unscheduled attribute on our programming MOOCs by using two quantitative variables: the instance pass rate and the average grade. Moreover, we observe the influence of the attribute above on the gradual progress over the time between scheduled and unscheduled instances. Altogether, we observed a total of five course instances. In the first two, the programming language was Java (one scheduled, one unscheduled instance), and the remaining three were in the Python language (two scheduled, one unscheduled). The objective was to try to find out if one of the methodologies has clear advantages for learning performance.

The rest of this paper is organized as follows. Section 2 reviews related work on technologies in programming education, programming MOOCs, and scheduled vs unscheduled instances of MOOCs. In Section 3, we concisely describe the history of MOOCs and the MOOC environment used at the University of Helsinki. The setup for this study is described in Section 4, and Section 5 gives the results of our quantitative experiments. In Section 6 we discuss the study and its limitations before concluding the paper and sketching some future directions to be studied in Section 7.

2. Related Work

2.1. *Technology in Programming Education*

The utilization of technology in programming education is widespread. In their systematic review of approaches for teaching introductory programming, Vihavainen *et al.* (2014) listed specific interventions. Of these, at least content change, game theme, and media computation are typically connected to the use of educational technology. As discussed in Cheah (2020), traditional materials and teaching methods are not efficient in programming education – creative use of technology support can adjust better to the dynamical nature of programming.

Automatic assessment is one of the most typical ways to utilize technology in programming education. A standard approach, see e.g. Bey *et al.* (2018), is to run student-submitted code against a set of predefined unit tests and determine if the results were expected. This or a similar approach is used in various learning environments (Ihantola *et al.*, 2010). The downside of the automatic assessment is often the quality of the feedback. For example, in Gordillo (2019), a significant number of students found the automated feedback challenging to understand and that the feedback is ‘of little use’.

Gamification is a concept that is typically connected to technology (although one can utilize gamification itself without technology). Lindberg *et al.* (2019) cites numerous studies about gamification in programming education and state that it can provide positive effects for learning, fun, and immersion. Moreover, as stated by Olsson *et al.* (2015), gamification and visualization can improve learners’ control and motivation and prevent feelings of loneliness, confusion, and boredom. Furthermore, gamification can also improve student engagement (Rojas-López *et al.*, 2019).

2.2. *Programming MOOCs*

In Reich and Ruipérez-Valiente (2019), the authors stated that MOOCs promised a new future for education but have faced some difficulties. The low retention rate is one of the most prominent ones. In their literature study, Khalil and Ebner (2014) found it to be less than 10%, estimating something as low as 7.5%[3]. Similarly, Onah *et al.* mentions that although drop-out rates seem to be high, the numbers are lower when considering students who complete some parts of the MOOC but do not intend to finish all the required parts.

According to Lepp *et al.* (2017), it is possible to increase the pass rate in a programming MOOC by activating students frequently, for example, by sending them weekly messages. It should be noted, however, that the number of participants in their course was only 32[4]. Personal activation in larger courses may not be as feasible. Although motivation has a significant role in completion, Luik *et al.* (2018) show that the students can be classified based on the *type of the motivation*, and not all motivation leads to likely completion (Luik and Lepp, 2021).

2.3. *Scheduled or Unscheduled?*

In unscheduled MOOCs, the students should be able to self-regulate and schedule their learning, which may be problematic for some learners (Littlejohn *et al.*, 2016). It is common to emphasize self-regulation in MOOC design (Littlejohn and Milligan, 2015; Pérez-Álvarez *et al.*, 2017). However, even recommending well-known self-regulating learning strategies does not seem to improve the course persistence (Kizilcec *et al.*, 2016). Hence, it seems that the students who can set their goals well can achieve the desired results (Handoko *et al.*, 2019).

[3] However, these are typically ‘evaluated without accounting for student intentions’ Reich (2014).

[4] One could argue whether this is a *massive* open online course.

Unscheduled MOOCs may also be problematic for universities as they provide ‘ongoing enrollment and completion outside traditional semesters’ (Kinash, 2013). Wild and Gimbrère (2017) state that some of the problems caused by this can be solved by combining ‘traditional’ course elements (such as on-campus exams) with the flexibility of MOOCs, but this likely makes the completion of the MOOCs more complex for many students. While many MOOCs provide real-time and any-time activities (Kinash, 2013), unscheduled MOOCs should give enough flexibility for students to define their schedules. A model combining the best of both worlds could be the future of education (Tsai, 2013).

Ihantola *et al.* (2020) performed a similar study in 2020 and found that the drop-out rates seemed to be concentrated to the beginning of the MOOC courses. They argue that mainly because of this, the retention rate in courses with deadlines is significantly higher than in courses without deadlines. According to the authors, the initial investment done in the early stages of the course leads to better success in completing the course.

3. Programming MOOCs at University of Helsinki

The University of Helsinki has successfully implemented Programming MOOCs (citation removed for the review process). In addition to providing good content, the focus has been on requiring the participants to be more active and get more assistance in their online learning. This has included support mechanisms such as workshops and tutoring sessions. The general idea has also been to blur the line between MOOC courses and traditional university courses by offering the same content and support mechanisms for all students regardless of their background.

Our programming MOOCs use the Test My Code (TMC) assessment service (Vihavainen *et al.,* 2013b). Initially, it was constructed for Java programming, but a later extension supports any programming language. TMC allows students to test their programming code against a series of unit tests enabling them to complete the tasks independently regardless of time and location. The tasks can be integrated into HTML pages, but especially later in the course, TMC is embedded into a programming IDE such as Net-beans (Bock, 2009) or Visual Studio Code (Del Sole, 2018). In addition to programming exercises, our MOOCs typically contain complementary exercise types such as quizzes.

The MOOCs are designed and implemented on a custom-made, open-source platform developed onsite at the University of Helsinki. The content is written with Markdown language and provided as standard HTML pages without any extra plugins (except for the additional software used to write, compile and test the programming tasks). Our MOOCs utilize gamification in some forms, including, for example, color-coding the exercise frames and providing a visualization of progress and score. The opening of subsequent modules based on the success of earlier modules can also be seen as a gamified element.

Java MOOC was the main programming MOOC offered by the University of Helsinki until 2020. The MOOC was divided into two parts: the first part, Introduction to

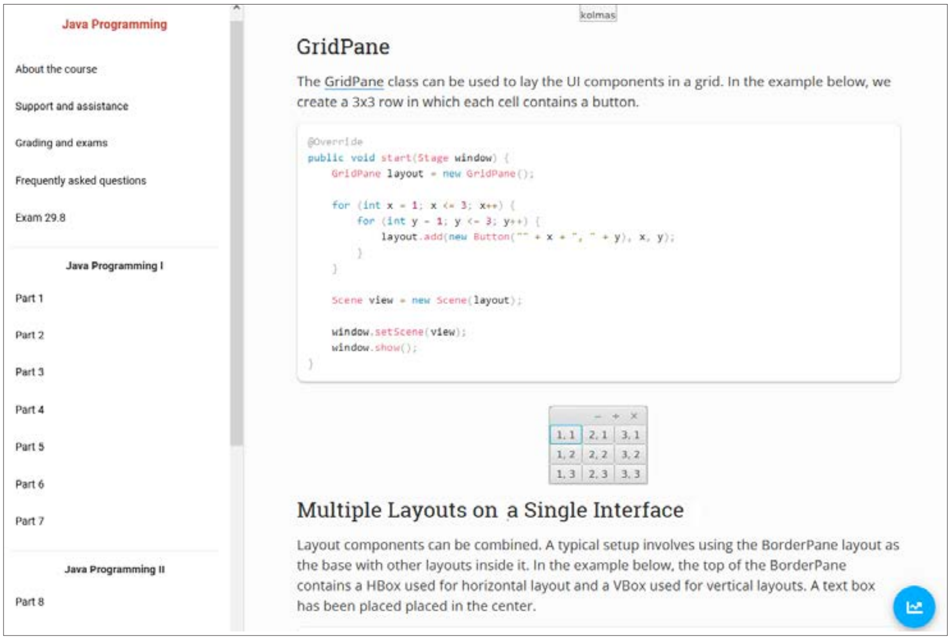
Programming, requires no initial programming knowledge. It covers the imperative paradigm (Watt and Wong, 1990) with some basic concepts of object-oriented programming. The second part, Advanced Programming, continues with the more advanced topic of object-oriented programming and also introduces concepts such as streams, regular expressions, and building graphical user interfaces. Both parts contain seven modules making the whole MOOC 14 modules long.

In 2020, at the University of Helsinki, Java was substituted by Python as the first programming language. The Java MOOC is still offered as a legacy course, but one can complete no official credits since the beginning of 2021. The Python course is similarly divided into two parts. The introductory part is designed with two goals in mind:

1. To provide good enough knowledge for anyone to use programming as a valuable tool to complete tasks like manipulating a CSV file or analyzing data.
2. To offer a solid foundation to build advanced programming knowledge.

The second part introduces more advanced Python concepts, such as object-oriented programming and inheritance, list comprehension, recursion, generators, and functional programming. The latter part is concluded by designing and implementing a computer game using the Pygame (McGugan, 2007) module. Similarly to the Java version, the Python MOOC contains 14 modules.

Fig. 1 shows a sample view of the Java MOOC. We applied similar design principles to both Python and Java MOOCs. There are very few videos or animations utilized. In-



GridPane

The GridPane class can be used to lay the UI components in a grid. In the example below, we create a 3x3 row in which each cell contains a button.

```
@Override
public void start(Stage window) {
    GridPane layout = new GridPane();

    for (int x = 1; x <= 3; x++) {
        for (int y = 1; y <= 3; y++) {
            layout.add(new Button("" + x + ", " + y), x, y);
        }
    }

    Scene view = new Scene(layout);

    window.setScene(view);
    window.show();
}
```

Multiple Layouts on a Single Interface

Layout components can be combined. A typical setup involves using the BorderPane layout as the base with other layouts inside it. In the example below, the top of the BorderPane contains a HBox used for horizontal layout and a VBox used for vertical layouts. A text box has been placed placed in the center.

Fig. 1. Example view from Java MOOC. We applied similar design and layout principles to all MOOCs observed in this study.

stead, the content provided consists mainly of static images, text and code examples. The coding tasks (and occasional other tasks, such as quizzes) are embedded within other content. In the first three weeks of the Python MOOC, the exercises can be completed directly on the MOOC web page with no additional tools needed. Starting from week four, all exercises are done with an external editor. In the Java version, the external editor is used throughout the course.

In the *scheduled* versions of MOOCs, we opened one module each week. The module usually contained 20 to 40 exercises, which needed to be completed by a given deadline (usually within 7 to 14 days). After all seven modules of the first part were opened, there was a two-week break around the exam. The latter part, with the eighth module, opened after the break. In the unschedule*d* versions, there is a single, shared deadline for all modules at the end of the year. Typically, students must obtain at least 25% of the score in a module before being allowed to access the next module. The scheduled and unscheduled versions utilize the very same exercises and other content.

We offer the students the possibility to receive a certificate for completion of the MOOC course without participating in an exam. If the student, however, wishes to earn study credits they must complete an exam. Typically, each MOOC instance provided several chances to take the exam; the exact number varied between three to seven. Students could take as many exams as they wanted, and the highest grade was continuously recorded. The schedules for the exams were flexible: typically, students were allowed to take the exam whenever they wanted within a given date.

4. Research Setup

The main comparison in this study is made between two instances of the Java programming MOOC organized in 2020. The first, *scheduled* instance had a new round of exercises opened each week, and all rounds had to be completed before the first exam. However, the students could take one of the resit exams later, but we gave no additional chances for completing the exercises. The students taking the scheduled version could also apply to the University of Helsinki if they completed the course and the separate entrance exam well enough.

The second, *unscheduled* instance of the course was otherwise similar to the scheduled one, but there students were allowed to complete the exercises on their own schedule, the only effective deadline was the end of the year. The students could participate in any exam during the year. The students who failed to complete the scheduled version were allowed to join the unscheduled version, and all the points from completed exercises were transferred by request.

We observe also three additional Python programming course instances. Two of them were scheduled and organized similarly to the first observed Java instance. The third one was an unscheduled instance, arranged similarly to the unscheduled Java one.

All the instances observed in this study are displayed and named in Table 1.

Table 1
All instances of the programming MOOCs observed in this study with abbreviations to be used subsequently

Abbreviation	Language	Year	Type	N
J20S	Java	2020	Scheduled	3972
J20U	Java	2020	Unscheduled	7148
P20S	Python	2020	Scheduled	895
P20S(2)	Python	2020	Scheduled	945
P21U	Python	2021	Unscheduled	9998

Now, the research questions to be considered in this study are as follows:

RQ1. Does the course pass rate distinguish between scheduled and unscheduled instances?

RQ2. Does the choice between scheduled and unscheduled instances influence the average grade?

RQ3. Does the choice between scheduled and unscheduled instances influence the course progress and completion time?

The corresponding null hypotheses are:

NH1. There is no difference in pass rates between scheduled and unscheduled instances.

NH2. There is no difference in average grades between scheduled and unscheduled instances.

NH3. The choice between scheduled and unscheduled instances does not affect course progress and completion time.

4.1. *Participants*

All the MOOC instances under this study were fully open to anyone interested. The fall 2020 instance P20S(2) was primarily aimed at students in the Bachelor's Programme in Computer Science at the University of Helsinki. However, all participants with other majors or outside the university were still welcome to take the course. To ensure fair and unbiased treatment, the course staff saw no indicators of students' backgrounds[5]. All materials were similarly open to all students, and the virtual lectures were linked to the course material.

The total number of students that started the MOOC instances under the study is listed in Table 2. A student was considered registered after starting the MOOC and completing at least one exercise in the MOOC with full points. The table also includes a percentage estimate of university students in each instance. The percentage is an

[5] To be precise, they had access to the email addresses students had chosen to use in the course registration, which gives a hint about student statuses. However, the university students were not obliged to use the university email address.

Table 2
The number of students starting each MOOC instance observed in this study and the estimated percentage of university students in each instance

Course	N	Uni. students%
J20S	3972	7.23%
J20U	7148	6.04%
P20S	895	22.01%
P20S(2)	945	34.71%
P21U	9998	11.38%

estimate because it is based only on students' email addresses. Although university students were encouraged to use their University address when registering, there are likely to be some exceptions.

4.2. *Procedure*

To see whether there are differences in student performance between instances, we focus on the percentage of students completing the exam and the average grade achieved. The first is measured by using **pass rate**: the number of students passing the course exam per the number of students considered registered (see above) at a given point in time, represented in percentage. In a scheduled instance, the number of students is simply the number of registered students. For the unscheduled case, however, this is somewhat more complicated as the students could jump in at any point in the course. To this end, in each unscheduled instance, we measured the number of students (and the pass rate based on that number) at two fixed points: after the first exam and at the end of the course. Although the total number of exams varied between the instances, the students had at least three chances to take the exam in all the instances under this study.

There were some cases where a student attended more than one observed instance. Especially, it was not unusual that a student first tried a scheduled MOOC instance and, after failing, continued with the corresponding unscheduled one. In such cases, the student participated in both instances but, in the corresponding scheduled instance, was considered a failed participant.

4.3. *Materials*

We described the materials used in the MOOC instances in Section 3. In this study, we observed the first seven modules of the instances corresponding to the course Introduction to Programming. Our instance evaluations are based on the students' performance in exercises (50% of the grade) and exam scores (50% of the grade). The exams were first automatically assessed, but course staff manually graded all answers that did not receive full points. The minimum grade (1) was assigned if a student achieved at least 50% of the total points. The maximum grade (5) required at least 90% of the total score.

The exams in the Java course instances (J20S and J20U) were shared, i.e., the students took the same exam regardless of whether they took the scheduled or unscheduled instance. Since the students could move from the scheduled instance to the unscheduled one, it is possible that they had points from both instances. In such a case, we took the instance with the highest score and considered that instance the one that the student passed. A total of 785 students participated in both instances.

In all the instances, the exams had three programming problems. The problems varied somewhat between Python and Java instances mainly because of language-dependent issues [6]. However, we did our best to make the difficulty level the same. The students were free to use any available materials during the exam (including the course material), but collaboration and communication were prohibited. If we detected any plagiarism, the exam was considered failed.

5. Results

In this section, we present the results of the study to the three research questions, RQ1 to RQ3. To that end, let us denote by N' and N the total number of students in an instance 1) when the first course exam took place and 2) at the end of the course, respectively. Accordingly, we denote by n' and n the number of students, when the first course exam took place and at the end of the course, and who successfully completed the course, respectively. Accordingly, we use R' and R for pass rates, and $\overline{G'}$ and $\overline{G}$ for average grades (when the first course exam took place and at the end of the course, respectively).

5.1. *Pass Rates*

The results of our study for **RQ1:** *Does the course pass rate distinguish between scheduled and unscheduled instances,* are gathered and displayed in Table 3.

Our first observation is that the pass rates of the scheduled instances are consistently and significantly higher than those of the unscheduled instances. The highest pass rate appears with the scheduled fall Python MOOC instance (P20S(2)). This is probably due to the higher rate of university student participants than in the other instances.

Another interesting observation is that after the first course exam, the pass rates of the scheduled instances are already almost at the level of the final pass rates. In contrast, the pass rates of the unscheduled instances clearly increase over time. There are two reasons for this. First, as revealed by the difference between N' and N, only a tiny portion of the students taking the unscheduled MOOCs had started when the first course

[6] For example, the absence of objects in the Python course.

Table 3

The pass rates of all course instances under consideration. For the unscheduled instances, N' and N denote the number of students, and R' and R are the pass rates after the first exam and at the end of the course, respectively. For the scheduled instances, N' and N simply correspond to the number of registered students, R' and R are interpreted as in the unscheduled case

Instance	N'	R'	N	R
J20S	3972	27.69%	3972	31.62%
J20U	2113	2.08%	7148	9.72%
P20S	895	19.22%	895	33.18%
P20S(2)	945	39.37%	945	44.34%
P21U	3908	6.00%	9998	11.41%

exam occurred. Moreover, the participants of the unscheduled instances tend to take a later edition of a course exam or take more than one of them to complete the course (as to be discussed in Section 5.3).

5.2. *Grade Averages*

Next, let us present our results for the research question **RQ2:** *Does the choice between scheduled and unscheduled instances influence the grade average*. The final grade averages of all course instances are displayed in Table 4. The grades run on a scale of 1 to 5, 5 being the best grade.

In this case, we ran a two-tailed T-test between the grade distributions of all instances (Table 5).

The only statistically significant difference ($p < 0.05$) was found between P21U to all other instances; the differences between other instances were not significant. Again, results after the first exam seem to be on par with the final results with the scheduled instances, while there were changes in both directions with the unscheduled instances. This is accounted for by an order of magnitude difference between the sample sizes at the time of measurements (i.e., between n' and n).

Table 4

The final grade averages of all course instances on a scale of 1 to 5. n' and n denote the number of students completing the course instance, $\overline{G'}$ and $\overline{G}$ the final grade averages after the first exam and at the end of the course, respectively

Instance	n'	$\overline{G'}$	n	$\overline{G}$
J20S	1121	4.47	1280	4.44
J20U	47	4.28	714	4.51
P20S	178	4.49	306	4.47
P20S(2)	381	4.47	429	4.46
P21U	193	4.80	1138	4.67

Table 5
The resulting p-values of the two-tailed T-test between the grade distributions of all instances

	J20s	J20U	P20S	P20S(2)
J20U	0.115			
P20S	0.75	0.40		
P20S(2)	0.75	0.31	0.96	
P21U	0.003	0.008	0.01	0.002

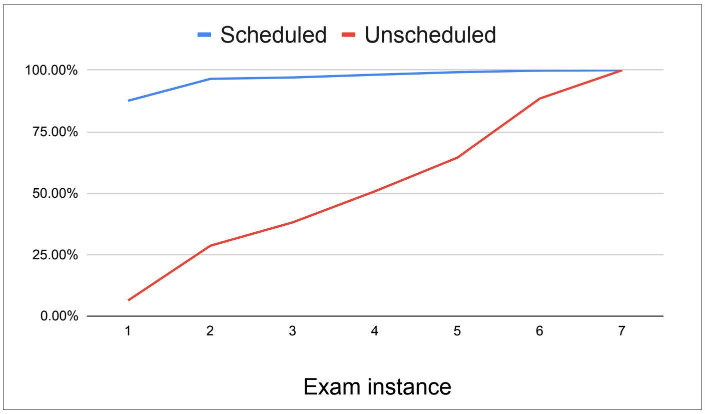


Fig. 2. The cumulative distribution function of course completion percentage in scheduled (blue line) and unscheduled (red line) Java course instances after each of the seven available exams.

5.3. *Completions over Time*

For the research question **RQ3:** *How does the choice between scheduled and unscheduled instances influence course progress and completion time*, we produced the time series displayed in Fig. 2. It depicts the number of students who have passed the course after each exam of the scheduled and unscheduled Java course instances. As shown in the figure, the progress is much slower in the unscheduled instance. Although there were seven exams in both courses (and all students were allowed to take any -or all -of them), the students in the scheduled instances seemed to prefer to complete the course as soon as possible, having done all the required exercises.

6. Discussion

Based on our study, the scheduled versions of MOOCs lead to significantly higher pass rates than the unscheduled ones, so we reject the first null hypothesis, NH1. All the scheduled instances had pass rates of over 30% (Table 3), while both unscheduled versions had significantly lower pass rates. In our study, the low pass rates of unscheduled versions follow the trend found in other studies (Onah *et al.*; Khalil and Ebner, 2014). We also observed that the number of university students attending the course affects the pass rate positively.

In average grades of the studied course instances, the only salient difference was in the unscheduled Python MOOC. It had a statistically significantly better average grade than all the other course instances. The content of the Python MOOC may be more suitable to be completed without a tight schedule. Regardless, the answer to our second research question seems unclear, so we are hesitant to reject the second null hypothesis, NH2.

Fig. 2 shows a significant difference between scheduled and unscheduled instances. With the scheduled instances, almost all completions are scored after the first two exams, while the progress is slower and steadier in the unscheduled version. This is quite logical as the students can start the course whenever they want and finish it when they feel like it. The total number of attendees also seems to be the highest in the unscheduled instances. Again, this seems evident as the number of attendees in the unscheduled Java course has more than tripled (from 2113 to 7148) between the first and the final exam. Altogether, there is a clear difference in the course progress and completion time favouring the scheduled instances over the unscheduled ones, and we can reject the third null hypothesis NH3.

The relatively high pass rates in scheduled instances seem to contradict the findings of other studies, see, e.g. Khalil and Ebner (2014). The known problems in self-regulation (Littlejohn *et al.,* 2016) may be concentrated more on the unscheduled versions, as the students with better self-regulation skills could be more likely to attend and pass the scheduled versions. Moreover, the weekly activation mechanism suggested by Lepp *et al.* (2017) can be seen as built-in in scheduled MOOCs, as the deadlines provide external motivation for timely completion. However, the unscheduled MOOCS may have other benefits for learning, as noted by Watson *et al.* (2018). Regardless, it is clear that scheduling and deadlines affect student performance and learning in MOOCs -something that has not been studied much before.

6.1. *Implications for Practice*

The results can help anyone implementing or studying MOOCs to understand the benefits and drawbacks of scheduling better. All in all, while both versions have their advantages, scheduled MOOCs seem more favourable, especially when observing the pass rates. However, the unscheduled MOOCs have a steadier stream of participants joining and completing the course. Hence, the choice should be made based on the institution's requirements, and we cannot provide an exhaustive answer on the superiority.

A natural suggestion would be to offer both versions simultaneously, whenever feasible. Based on the results, this may give the best of both worlds. However, providing two different versions of the same MOOCs would require more work and can lead to scheduling and materials problems. For example, correct answers for the scheduled

MOOC cannot be published while students still work on the tasks in the unscheduled version. Moreover, it is unclear whether the students would select a scheduled instance over an unscheduled one if no external motivation[7] is provided.

6.2. *Study Limitations*

There are some limitations of the study that have to be considered. Most importantly, the scheduled Java MOOC (J20S) also served as a possible method for applying to the Bachelor's Programme in Computer Science at the University of Helsinki. It is impossible to determine the number of students who took that instance solely to be admitted to the university. Still, it is likely that the number of attendees in the scheduled instance would have been smaller if there had not been such an option. However, the two other scheduled instances seem to provide similar pass rates and grade averages. Thus, the most significant effect was likely on this case's total number of students.

6.3. *Future Research*

In the future, we would like to study the methodological differences in instances of different kinds of MOOCS, including other computer science MOOCs, but also courses on any other topic. We would also like to include a more specific evaluation of students' engagement with MOOCs and whether there are differences in when and how they are engaged in scheduled or unscheduled MOOCs (see, e.g. the findings in Ihantola *et al.* (2020)). It would also be beneficial to include students' perceptions about their motives for taking the MOOC, their preferred methodology, and what variables affect this. For example, some students likely do not even want to complete the MOOC but instead join to learn one or more individual skills.

7. Conclusion and Future Work

We compared the differences between scheduled and unscheduled instances of programming MOOCs. We found out that while the pass rates are significantly higher in scheduled instances, there seem to be little differences in grade averages. For the latter, however, the unscheduled Python MOOC stood out as an exception: the average grade of it was significantly higher than in all other observed instances. Moreover, it seems that the number of university students participating in the course positively affects the pass rate. Finally, in the scheduled versions, the students tend to attend a course exam swiftly, completing the exercises. In contrast, the students of unscheduled instances join and complete the course throughout the observed period. However, the unscheduled instances lead to more participants and completions.

[7] Such as admittance to university.

In this study, we measured the success of the courses based only on two quantitative variables: pass rate and average grade. Although they are widely used and can be considered valid measurements for course success, there are other factors that one should consider in the future. For example, student satisfaction and perceptions and self-regulatory strategies should be measured and considered. The students may react differently to unscheduled and scheduled versions regarding motivation, pressure, and stress, to name a few. In the future, we will extend our study to other types of computer science MOOCs and MOOCs of different disciplines.

References

Bartolomé, A., Steffens, K. (2015). Are MOOC Promising Learning Environments? (preprint copy). *Comunicar*, 44. `https://doi.org/10.3916/C44-2015-10`

Bey, A., Jermann, P., Dillenbourg, P. (2018). A comparison between two automatic assessment approaches for programming: An empirical study on MOOCs. *Journal of Educational Technology & Society*, 21(2), 259–272.

Bock, H. (2009). *The Definitive Guide to NetBeans Platform*. Apress, Berkeley, CA.

Cheah, C.S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, 12(2), 272.

Del Sole, A. (2018). *Visual Studio Code Distilled: Evolved Code Editing for Windows, MacOS, and Linux*. Apress, Berkeley, CA.

Gordillo, A. (2019). Effect of an instructor-centered tool for automatic assessment of programming assignments on students' perceptions and performance. *Sustainability*, 11(20), 5568.

Handoko, E., Gronseth, S.L., McNeil, S.G., Bonk, C.J., Robin, B.R. (2019). Goal setting and MOOC completion: A study on the role of self-regulated learning in student performance in massive open online courses. *International Review of Research in Open and Distributed Learning*, 20(3).

Ihantola, P., Ahoniemi, T., Karavirta, V., Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. In: *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, pp. 86–93.

Ihantola, P., Fronza, I., Mikkonen, T., Noponen, M., Hellas, A. (2020). Deadlines and MOOCs: How do students behave in MOOCs with and without deadlines. In: *2020 IEEE Frontiers in Education Conference (FIE)*, IEEE, pp. 1–9.

Khalil, H., Ebner, M. (2014). MOOCs completion rates and possible methods to improve retention-A literature review. In: *EdMedia+ Innovate Learning*, pp. 1305–1313. Association for the Advancement of Computing in Education (AACE).

Kinash, S. (2013). MOOCing about MOOCs. *Education Technology Solutions*, 57(70), 56–58.

Kizilcec, R.F., Pérez-Sanagustín, M., Maldonado, J.J. (2016). Recommending self-regulated learning strategies does not improve performance in a MOOC. In: *Proceedings of the third (2016) ACM conference on learning@ scale*, pp. 101–104.

Kovanovic, V., Joksimovic, S., Gasevic, D., Siemens, G., Hatala, M. (2015). What public media reveals about MOOCs: A systematic analysis of news reports. *British Journal of Educational Technology*, 46, 510–527.

Lepp, M., Luik, P., Palts, T., Papli, K., Suviste, R., Säde, M., Tõnisson, E. (2017). MOOC in programming: A success story. In: *Proceedings of the International Conference on e-Learning*, pp. 138–147.

Lindberg, R.S., Laine, T.H., Haaranen, L. (2019). Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games. *British Journal of Educational Technology*, 50(4), 1979–1995.

Littlejohn, A., Milligan, C. (2015). Designing MOOCs for professional learners: Tools and patterns to encourage self-regulated learning. *eLearning Papers*, 42.

Littlejohn, A., Hood, N., Milligan, C., Mustain, P. (2016). Learning in MOOCs: Motivations and self-regulated learning in MOOCs. *The Internet and Higher Education*, 29, 40–48.

Luik, P., Lepp, M. (2021). Are highly motivated learners more likely to complete a computer programming MOOC? *International Review of Research in Open and Distributed Learning*, 22(1), 41–58.

Luik, P., Lepp, M., Palts, T., Säde, M., Suviste, R., Tõnisson, E., Gaiduk, M. (2018). Completion of programming MOOC or dropping out: Are there any differences in motivation. In: *Proceedings of the 17th European Conference on e-Learning ECEL*, pp. 329–337.

McGugan, W. (2007). *Beginning Game Development with Python and Pygame: From Novice to Professional*. Apress, Berkeley, CA.

Olsson, M., Mozelius, P., Collin, J. (2015). Visualisation and gamification of e-learning and programming education. *Electronic Journal of e-Learning*, 13(6), 452–465.

Onah, D.F., Sinclair, J., Boyatt, R. Dropout rates of massive open online courses: Behavioural patterns.

Pérez-Álvarez, R., Maldonado-Mahauad, J.J., Sapunar-Opazo, D., Pérez-Sanagustín, M. (2017). Note-My-Progress: A tool to support learners' self-regulated learning strategies in MOOC environments. In: *European Conference on Technology Enhanced Learning*, Springer, pp. 460–466.

Petkovska, B., Delipetrev, B., Zdravev, Z. (2014). MOOCS in Higher Education? State of the Art Review. In: *International Conference on Information Technology and Education Development*, pp. 108–112.

Pirttinen, N., Leinonen, J., Lemström, K. (2021). Digital education for all: better students through open doors? In: *ITiCSE 2021: 26th ACM Conference on Innovation and Technology in Computer Science Education*, pp. 450–456.

Reich, J. (2014). MOOC completion and retention in the context of student intent. *EDUCAUSE Review Online*, *8*.

Reich, J., Ruipérez-Valiente, J.A. (2019). The MOOC pivot. *Science*, 363(6423), 130–131.

Rodriguez, C.O. (2012). MOOCs and the AI-Stanford like courses: Two successful and distinct course formats for massive open online courses. *The European Journal of Open, Distance and E-Learning*, 15.

Rojas-López, A., Rincón-Flores, E.G., Mena, J., García-Peñalvo, F.J., Ramírez-Montoya, M.S. (2019). Engagement in the course of programming in higher education through the use of gamification. *Universal Access in the Information Society*, 18(3), 583–597.

Sanchez-Gordon, S., Luján-Mora, S. (2016). How Could MOOCs Become Accessible? The Case of edX and the Future of Inclusive Online Learning.

Tsai, F.S. (2013). The State of Massive Open Online Courses (MOOCs) in Engineering Education: Where do we go from here? In: *American Society for Engineering Education* (Vol. 23), pp. 1–16.

Vihavainen, A., Airaksinen, J., Watson, C. (2014). A systematic review of approaches for teaching introductory programming and their influence on success. In: *Proceedings of the Tenth Annual Conference on International Computing Education Research*, pp. 19–26.

Vihavainen, A., Luukkainen, M., Kurhila, J. (2013a). MOOC as semester-long entrance exam. In: *Proceedings of the 14th Annual ACM SIGITE Conference On Information Technology Education*, pp. 177–182.

Vihavainen, A., Vikberg, T., Luukkainen, M., Pärtel, M. (2013b). Scaffolding students' learning using test my code. In: *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, pp. 117–122.

Watson, W.R., Yu, J.H., Watson, S.L. (2018). Perceived attitudinal learning in a self-paced versus fixed-schedule MOOC. *Educational Media International*, 55(2), 170–181.

Watt, D.A., Wong, S. (1990). Programming Languages. *Concepts and Paradigms Prentice Hall*.

Wild, U., Gimbrère, M. (2017). MOOCs: Introducing Flexibility in Academia. In: *EMOOCs-WIP*, pp. 133–138.

E. Kaila received his Ph.D. in Computer Science in 2018 from University of Turku. His current research interests include programming education, educational technology, learning analytics and ethics in digital learning.

K. Lemström received his Ph.D. in Computer Science in 2000 from the University of Helsinki, where he now is a Senior University Lecturer. His current research interests include algorithmics, music informatics and learning analytics.