Assessing computer program reading comprehension skill in pre-service teachers: The development and piloting of a screening instrument in Scratch

Eva Marinus, Michael Hielscher, Beat Döbeli Honegger
Pädagogische Hochschule Schwyz
eva.marinus@phsz.ch

Abstract
The recent addition of computer science and computer programming to primary school curricula worldwide introduces the challenge of bringing primary school teachers up to speed in this new learning domain. Assessments are needed to determine their knowledge and competencies in this area. The aim of this study was to develop and pilot a program reading comprehension test (the "SRCT") in Scratch, a frequently used programming language in primary schools. The test consists of 12 items that can be administered within 30 minutes via a Google Forms interface. We piloted the test with two cohorts of Swiss, German-speaking pre-service teachers. The first cohort (n=94) filled out the test during class at the end of semester and the second cohort (n=74) from home and at the beginning of semester. In both contexts, the test showed an internal consistency of .68. Such internal consistency is considered acceptable for a screening instrument and removal of items neither consistently (between the samples), nor substantially increased the internal consistency. We found no significant differences in performance between men and women. We also examined validity by comparing the performance pattern on the test with that on the Scratch exam assignment. However, more specific research, with one-on-one links between individual test performance and a larger variety of tests are needed to draw conclusions about this aspect of the SCRT.

Keywords
Computer science education; programming assessment; pre-service teachers; Scratch.

Across the globe, awareness is growing that the next generation needs to be prepared for the Digital Transformation, and worldwide changes in the curricula have been made accordingly (Balanskat & Engelhardt, 2015). In Switzerland, the German speaking cantons have done this by introducing the learning domains "Medien und Informatik" in Lehrplan-21 (kindergarten to Grade 9) in 2015 (D-EDK, 2016). Due to these curriculum adaptations, in-service and pre-service teachers need to expand their skills to confidently teach these new learning domains. Results from both national and international studies show that there is still a lot of work to do: pre-service teachers both lack knowledge and hold many misconceptions about computer science topics like programming, computer systems, and data (Döbeli Honegger & Hielscher, 2017; Yadav & Berges, 2019). Remediating this knowledge gap is not easy, as most current pre- and in-service primary school teachers did not have or barely had the opportunity to learn computer science skills, like programming, during their own education. This means that, before they can learn how to teach, teachers first need to become proficient programmers themselves. This is very different from the situation in other learning domains, like mathematics and literacy, in which they are already skilled and confident. How proficient future teachers need to be in each learning domain is still a topic of debate and different levels have been suggested (e.g., Krauss et al. 2008). However, research has demonstrated that the ability to teach a topic will increase, when the teacher has better competencies in the topic at hand (e.g., Tröbst et al., 2018).
The introduction of new learning domains to the curriculum also brings the challenge of assessing corresponding competencies. This is not only the case for the assessment of students in school, but also for the assessment of the teachers who are in training. The aim of this study is

to pilot a screening instrument for programming ability in Scratch (Resnick et al., 2009). Specifically, this instrument aims to assess the ability to read and understand Scratch programs, as this is required from teachers, when they are helping their students during programming classes. Moreover, reading and understanding of programming is often mentioned as the first step in learning to program (e.g., PRIMM Model, Sentence et al., 2019). Up until now, most efforts to assess performance in Scratch have focused on the evaluation of artefacts such as projects, programs, and portfolios. Such assessments are typically time consuming to score and interpret. Moreover, research has shown that even if students or novice programmers can generate working programs, it does not mean that they understand the underlying program concepts that make the program function (Salac et al., 2020). We therefore wanted to develop an instrument that directly and quickly measures reading comprehension of Scratch programs. Moreover, as far as we are aware, there are no psychometric tests for programming ability in Scratch available yet. We piloted our test with two cohorts of pre-service teachers during a compulsory computer science training at our university of teacher education. During the course these bachelor students completed guided tutorials and created programs in Scratch, but there was no focus on reading and interpreting existing programs that were similar to the SRCT questions. In other words, students were not trained with exercises like those presented in the SRCT. The only requirement to complete the test was that the students were familiar with the Scratch environment. The first cohort was tested at the end of the course, and we also examined if class performance on the Scratch test matched performance on the programming assessment that was part of the exam. The second cohort was tested at the beginning of the course and for this cohort we also analyzed potential performance differences between women and men.

Methods

Participants

Participants were two cohorts (2019 and 2020) of first-year bachelor students from a teacher education university in Switzerland. They were enrolled in a basics-of-CS course during which they also learned how to teach CS to primary school children. Of the 2019 cohort, 69.3% indicated that they never programmed before. For the 2020 cohort this was 63.7%. The students of each cohort were divided in four different subgroups, which were taught (two subgroups each) by the second and third authors. The 2019 cohort consisted of 94 students (median age: 21, age range: 18-49, 72 women, 22 men. Of this cohort all 86 students who were present during the final lecture (12th week of semester) were requested to fill out the Scratch Reading Comprehension Test (SRCT) online test. Eighty-three of them gave permission to use their test results for this research. The 2020 cohort consisted of 90 students of which 80 filled out the online form and 74 of them (median age: 20, age range: 18-48, 60 women, 14 men gave permission for their data to be used for research. For this cohort, testing took place at the beginning of the course (3rd week of semester) and was done as part of their individual learning at home.

Materials

Scratch Reading Comprehension Test (SRCT)

The test item programs were inspired by projects that were saved online by the students that were enrolled in the course in 2019 and by example programs from Code Club (codeclub.org/en/). For each item we presented a Scratch stage with a sprite and a description of what the program was doing. The task of the participants was to pick one of four pieces of Scratch code that matched the description. The other three options were incorrect. As all four options consisted of the same Scratch blocks, the test can be seen as a simplified (multiple choice) version of easy Parsons Problems (Denny et al., 2018).

During test construction, we considered two additional factors. 1. Striking a balance between having a fair number of items and still being able to administer the test in 30 minutes. This led to the decision to include 12 items, which were presented in the same order to all students. 2. Finding a balance between program length (to not overload the working memories of the students)

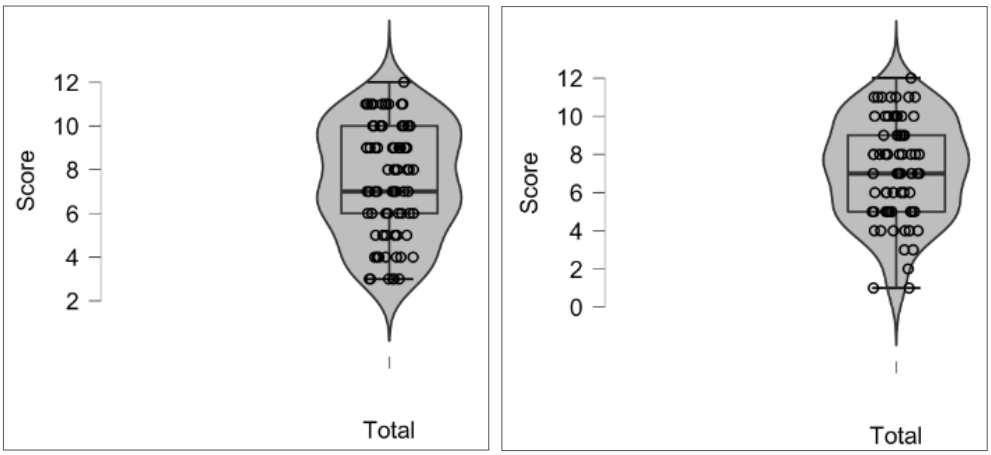
and having enough relevant blocks to create programs that were still meaningful (but incorrect) when putting the blocks in another order. We did not put in a specific effort to create items of different difficulty, nor did we try to rank them from easy to hard because we did not have any specific hypotheses about item difficulty at the start of this research. A PDF of the test can be found on the Open Science Framework (OSF) (https://tinyurl.com/uj8ndert).

Scratch Programming during the course and the Scratch assignment of unit exam
A description (in German) of the course can be found here: https://mia.phsz.ch/GDI19. Programming in Scratch was covered during weeks 2-5 of the course. The students practised programming with guided tutorials (group puzzle setting) and later in a pair programming setup. As part of the exam, the students had to write a program in Scratch (e.g., a computer game or simulation). An important requirement was that the program had to go beyond animation: the user should be able to interact with it. The students were working in pairs using the pair programming method. For a full description of the assignment (in German) see here: https://mia.phsz.ch/GDI19/ModulPruefung.

Procedure
Testing of the 2019 cohort took place at the teacher education university on a morning in December 2019 during the last 30 minutes of the class. The test was introduced by the first author who explained the goal of the exercise ("to find out how well you can read Scratch programs and to improve the unit contents") and stressed that the test was not part of the exam, however, that it could be a useful self-test in preparation for the final exam. They were also instructed that participation was voluntary and anonymous, that they could tick a box to opt out to have their data used for research and that it was important that they only filled out the test once (to avoid getting more than one data set from the same person). Finally, the students were told how to start the test via a link and how to make sure that they ended it correctly so that the data was saved. The SRCT was presented online via Google Forms in German. It took the students between 10 and 25 minutes to fill out the test.
The testing of the 2020 cohort took place during the 3rd week of semester. Due to COVID19, the students received written instructions via the course website (see here https://mia.phsz.ch/GDI20/AlgorithmenTest) and filled out the SRCT at home within a two week timeframe (between the 17th and 28th of September), they could indicate in the online form, if they were happy for their data to be used for research.
To increase willingness for participation the data collection was anonymous. After submitting the results, participants of both cohorts could see how many items they answered correctly and the correct solution for the items they answered incorrectly. Skipped items were scored as wrong. The raw data of both cohorts can be found on the OSF, https://tinyurl.com/uj8ndert.

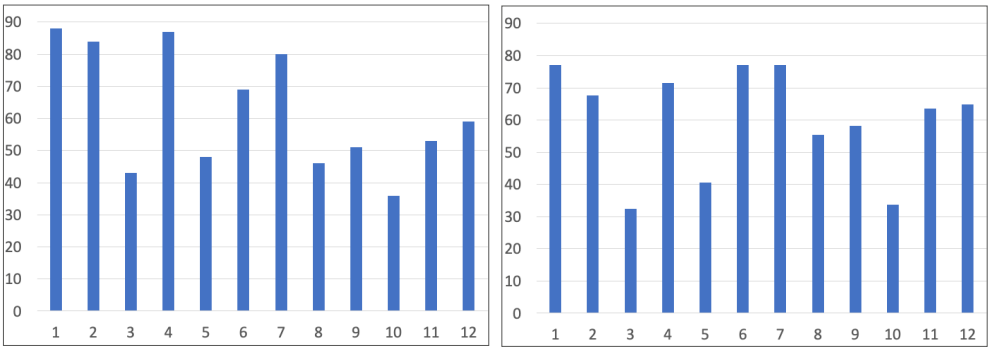
Results
First, we inspected the distribution of scores by creating separate violin plots for the two cohorts in JASP (2018) (See Figure 1). The distributions were very similar. For the 2019 cohort, the mean score was 7.4 out of 12 and the standard deviation 2.5. For the 2020 cohort, the mean score was 7.2 out of 12 and the standard deviation 2.6. For both samples, the data was approximately normally distributed, and there were no floor or ceiling effects. The minimum score was 3 for the first cohort and 1 for the second cohort and the maximum score for both cohorts was 12.

Figure 1: Distribution of scores on the SRCT for the 2019 (left) and 2020 (right) cohorts. Every circle represents a participant.



Second, we inspected performance per item by calculating the mean score for each of the twelve items. As can be seen in Figure 2, the results were very similar for both cohorts: item difficulty showed the same pattern. In addition, averages for all items were above 25%, suggesting that the majority of students did not simply guess the answers but put in at least some effort till the last item. For the 2020 cohort we also asked the sex of the participants. Using a one-way ANOVA, we found no significant differences between men (n=14) and women (n=60), F (1;72)=0.37, p=.55.

Figure 2: Percentage correct per item for the 2019 (left) and 2020 (right) cohorts



Finally, to determine if the items on the SRCT measure the same construct and therefore produce similar scores for the same participants, we calculated the internal consistency. For both cohorts, McDonald's Omega was 0.680 (95% CI for cohort 1: 0.58-0.78), for cohort 2: 0.57-0.79). When excluding each of the 12 items one by one, McDonald's Omega fluctuated between .614 and .687 for the first cohort and between .638 and .696 for the second cohort. For the first cohort, the removal of three items slightly increased the internal consistency: item 1 (.687), 2 (.684) and 10 (.686). When we removed these three items at once, McDonald's Omega was .700. For the second cohort, the removal of two items slightly increased the internal consistency: item 3 (.696) and item 5 (.695). When we removed these two items at once, McDonalds Omega was .714.

Because of the anonymity of our SRCT test it was not possible to directly link the individual module exam outcome of each student to their individual test performance on the SRCT. However, we could still compare the performance of the four classes on the module exam. For these analyses we did not use the 2020 cohort, as these students completed the test at their own time at home. First, we conducted a two-way ANOVA to investigate the effects of teacher (1 and 2) and time point of class (early morning 8-10 and late morning 10-12) on the outcome of the SRCT. As this

analysis was exploratory, we tested conservatively, setting the critical p-value at .016 (Bonferroni correction: 0.05/3). We found a significant main effect for the time point of class, $F(1; 79) = 8.67$, $p = .004$, $\eta_p^2 = .098$), with the students in the early classes scoring significantly higher (M = 8.13, SD = 2.29) than those in the later classes ($M = 6.49$, $SD = 2.55$). The main effect of teacher and the interaction effect between time point of class and teacher were not significant.

If Scratch comprehension skill, as measured with the SRCT, was similar to what was measured with the Scratch exam assignment, we would expect the same performance pattern. This was not the case: A two-way ANOVA (critical p-value at .016, time and teacher as fixed factors), showed that there was no significant difference between late and early morning. However, we found that the two classes of teacher 2 performed slightly (1.3 points on a max of 24), but significantly better than the two classes of teacher 1 on the Scratch exam assignment, $F(1; 87) = 15.95$, $p < .001$, $\eta_p^2 = .16$). The interaction between the time point of class and teachers was not significant.

Discussion

The aim of this study was to develop and pilot a screener for program reading comprehension in Scratch that can be used to quickly assess pre-service or in-service teachers with some experience in the Scratch environment. Because it was not explicitly developed with items increasing in difficulty, it does not give any information on programming ability level. The Scratch Reading Comprehension Test (SRCT) can be administered during class or from home. In both contexts, the test shows an internal consistency of .68. Rounded to .7 this is considered acceptable for a screening instrument and removal of items neither consistently (between the samples), nor substantially increased the internal consistency. In our 2020 sample we found no significant differences in performance between men and women.

We did not aim to evaluate the effectiveness of the course in terms of programming reading comprehension as his skill was not explicitly trained before testing. Indeed, our results show that there are no differences between the cohort of 2019 that completed the SRCT at the end of the unit and the cohort of 2020 that completed it at the beginning of the course. However, the results do show that there is a large variability of performance between the students: some of the students get (almost) everything right and some of them are at a guessing rate. Apparently, some students either start their course with experience with Scratch or are much faster than their peers to pick up programming reading comprehension skills.

The average of the populations was just above 7 out of 12 of the items. As there are no norms for the SCRT, this mean of 7 is hard to interpret. However, all 12 items consisted of relatively easy programs that all teachers should be able to read and understand. Because effective teaching of programming requires the ability to read and understand the programs the children in their classes are developing, we suggest that the instrument, even without norms, can already be used to identify students who do not show any reading ability or comprehension at all (scoring at or below guessing rate, <5 out of 12). These students should be encouraged to engage in additional practising of reading and understanding Scratch programs.

We also found that the performance of the students, who took the test in the early classes, was slightly better than that of those in the later classes. This pattern was not found for the Scratch assignment of their course exam. This result could be interpreted as an indication that the SRCT measures different skills than the Scratch assignment for the exam (divergent validity). However, it could also be that the students in the late classes were just performing worse as they completed the test just before lunch and might have been hungry. More targeted research, where relations between SCRT and performance on the other task is linked for each individual, is needed to answer questions about divergent and convergent validity of the SCRT. For the 2021 cohort we made a first step into this direction: we collected data on the SCRT, the exams and an adapted version of the Computational Thinking test (Román-González, 2015) and asked the students for permission to link the scores. The data is currently under analysis

Acknowledgements

We would like to thank the students who participated in the study. This study was financed by

the Swiss National Science Foundation (SNSF) as part of the National Research Programme NRP77 Digital Transformation, project no. 407740_187447.

References

Balanskat & Engelhardt (2015). Computing our future: Computer programming and coding Priorities, school curricula and initiatives across Europe. European Schoolnet (2015). http://www.eun.org/c/document

Denny, P., Luxton-Reilly, A., & Simon, B. (2008). Evaluating a new exam question: Parsons problems. Paper presented at the Proceedings of the Fourth international Workshop on Computing Education Research, Sydney, Australia. https://doi.org/10.1145/1404520.1404532

Deutschschweizer Erziehungsdirektoren-Konferenz (D-EDK), Hrsg. 2016. «Medien und Informatik.» In Lehrplan 21. Von der D-EDK Plenarversammlung am 31.10.2014, bereinigte Fassung vom 29.2.2016, 479–498. D-EDK-Geschäftsstelle.

Döbeli Honegger, B.; Hielscher, M. (2017). Vom Lehrplan zur LehrerInnenbildung – Erste Erfahrungen mit obligatorischer Informatikdidaktik für angehende Schweizer PrimarlehrerInnen. In INFOS 2017, E-Learning Fachtagung Informatik, Lecture Notes in Informatics (LNI). Proceedings Series of the Gesellschaft für Informatik (GI) Volume P-274, S. 97-107.

JASP Team. 2018. JASP (Version 0.8.6).

Krauss, S., Neubrand, M., Blum, W., Baumert, J., Brunner, M., Kunter, M., & Jordan, A. (2008). Die Untersuchung des professionellen Wissens deutscher Mathematik-Lehrerinnen und -Lehrer im Rahmen der COACTIV-Studie. Journal für Mathematik-Didaktik, 29(3–4), 233–258.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. (2009). Scratch: Programming for All. Commun. ACM 52, 11, 60–67. https://doi.org/10.1145/1592761.1592779

Román-González, M. (2015). Computational thinking test: Design guidelines and content validation. Proceedings of EDULEARN15 conference, july (2015), pp. 2436-2444. https://doi.org/10.13140/RG.2.1.4203.4329

Salac, J. & Franklin, D. (2020). If They Build It, Will They Understand It?: Exploring the relationship between student code and performance." 25th Annual Conference on Innovation and Technology in Computer Science Education. https://doi.org/10.1007/BF03339063

Sentance, S., Waite, J. & Kallia, M. (2019) Teaching computer programming with PRIMM: a sociocultural perspective, Computer Science Education, 29:2-3, 136-176.

Tröbst, S., Kleickmann, T., Heinze, A., Bernholt, A., Rink, R., & Kunter, M. (2018). Teacher knowledge experiment: Testing mechanisms underlying the formation of preservice elementary school teachers' pedagogical content knowledge concerning fractions and fractional arithmetic. Journal of Educational Psychology, 110(8), 1049–1065. https://doi.org/10.1037/edu0000260

Yadav, A. & Berges, M (2019). Computer science pedagogical content knowledge: Characterizing teacher performance. ACM Transactions on Computing Education, 19 (3). 1-24.