Programming concepts and misconceptions in grade 5 and 6 children: Developing and testing a new assessment tool

Marco Hartmann[a,b], Peter Edelsbrunner[b], Michael Hielscher[a], Giulia Paparo[a], Beat Döbeli Honegger[a], Eva Marinus[a]
[a] Pädagogische Hochschule Schwyz
[b] Eidgenössische Technische Hochschule Zürich
marco.hartmann@phsz.ch

Abstract
Many countries are implementing computer science education, including programming, into their school curricula. In primary school this topic is new, both for students and teachers. Even though there has been research into the difficulties that high school and adult students face when learning to program, not all these findings translate well to the cognitive abilities of younger students and the environment they learn in. Our research focuses on finding out which misconceptions about programming primary school children hold and develop. This is important because research from other fields has shown that knowledge about misconceptions can improve teaching. This project aims to develop a Programming Misconceptions Assessment Tool (ProMAT) for children in grades 5 and 6 that have learned to program using either xLogo or Scratch – two popular educational programming languages currently used in Switzerland. Ultimately, the project should result in an assessment tool that can be used both by researchers and practitioners interested in primary school children's misconceptions about programming. In addition, it should reveal if and how educational programming environments affect the development of these misconceptions. The ProMAT is currently under development and underwent a first pilot testing phase with 57 children. In this paper, we describe the development of the first version of this tool, discuss insights gained from the pilot study, and outline the next steps of the project.

Keywords
Misconceptions; Computer Programming; Primary School; Assessment; Scratch.

Introduction

Across the world there has been an increase in and a shift towards early Computer Science (CS) and programming education (Falkner et al., 2019; Passey, 2017). In the German-speaking part of Switzerland, the subject *Media and Informatics* is part of the official school curriculum, *Lehrplan 21*. Within this curriculum, programming is introduced as early as grade 5, with the stated goal of the children being able to write programs containing loops, conditionals, and parameters by the end of grade 6 (Deutschschweizer Erziehungsdirektoren-Konferenz, 2015). Similar curricula exist in the French- and Italian-speaking parts of the country. Preparing teachers for the task to teach this new domain is posing a significant challenge.

Computer Science as a University degree has been introduced for decades and there exists already a large body of accompanying education research (Luxton-Reilly et al., 2018). One area of focus in this field is the difficulties, including misconceptions, that students face when learning to program. In the context of programming, misconceptions can be defined as an incorrect understanding of a concept or a set of concepts that leads to making mistakes in writing or reading programs (Sorva, 2008). Research in other domains, such as physics (Edelsbrunner et al., 2018) or statistics (Iten, 2014), showed that learning about misconceptions can improve teaching a given subject. Research into misconceptions in the field of programming has mostly been focussed on adults in the past, therefore studies and assessments for children are still comparatively sparse.

To gain insight into what causes difficulties and misconceptions of learners, reliable assessment instruments need to be developed. As far as we are aware, only a few researchers have tried to assess misconceptions in children (Mladenović et al., 2018; Swidan et al., 2018; Žanko et al., 2019). Of these, only Swidan et al. (2018) developed an assessment specific for a wider range of programming misconceptions. In this project, we aim to create and validate a quantitative tool to measure programming misconceptions in grade 5 and 6 primary school children, and then use this tool to compare the impact of different programming environments on children's development of misconceptions. There exists a wide variety of educational programming languages that are specifically designed to facilitate the introduction of new and young learners. Two such programming environments currently used among others in Swiss primary schools, and for which teaching materials exist, are xLogo and Scratch. Due to their differences in features and design choices (e.g., block-based vs. text-based programming editors), we chose these two environments for our investigation and our assessment tool will be developed in versions for both environments.

Development of the Programming Misconception Assessment Tool (ProMAT)

To find relevant programming misconceptions, we started with a compilation of misconceptions collected by Sorva (2013), as well as a few misconceptions from more recent research focussing children (Grover & Basu, 2017; Rich et al., 2017). From this collection of 167 misconceptions, we excluded 127 that either focussed on object-oriented programming languages, were not implementable for the xLogo or Scratch programming languages, or involved concepts that we deemed too complex for the target population (e.g., higher-order functions, recursion). Three additional misconceptions were excluded because their description was unclear or the same idea was better represented in another misconception. The remaining misconceptions were sorted into six categories. These categories were chosen based on the curriculum, CS research literature, and proposed learning trajectories for K-8 children by Rich et al. (2017). Two categories (Variables and Functions) were excluded because they are not part of the *Lehrplan 21* curriculum for grades 5 and 6. The remaining four misconception categories, forming the basis for the development of the assessment instrument, were: Sequentiality, Loops, Conditionals, and Superbug (see Table 1). A full list of the currently included misconceptions can be found on the Open Science Framework page of this project (https://tinyurl.com/pf9pkrvf).

Table 1. Showing the four chosen categories and representative misconceptions as example.

Category	Description	Examples
Sequentiality	This is about understanding that a computer program is executing one statement after the other in a linear fashion (not considering more complex concurrent programming).	«The order of commands does not matter.»
Loops	These allow for a specific part of a program to be repeated.	«Adjacent code executes within loop.»
Conditionals	These allow for branching – a way to select which parts of a program are executed depending on specific conditions.	«Code after IF statement is not executed if the THEN clause is.»
Superbug	Describes the idea that a programming language or environment possesses humanlike intelligence with interpretive power. A concept first proposed by Pea (1986).	« The system does not allow unreasonable operations.»

Next, we developed items to test misconceptions in each of these four categories. In addition, we had the following three constraints in mind. First, it should be possible for most children to complete the assessment within 45 minutes, the typical duration of a lesson in Swiss primary schools. Second, for ease of use for teachers and to make analysis in larger populations possible, it needs to provide quantitative data that can be automatically evaluated. Finally, it needs to address our specific categories of programming misconceptions individually. Therefore, we opted for a multiple-choice test format with five tasks per category where participants choose one of four answer options for each task.

All test items show a short computer program and a question regarding the outcome of said program. As of now only the Scratch version of the ProMAT has been developed, the parallel xLogo version is planned in the next step. From early prototype testing it was evident that if children lack basic knowledge of and experience with the programming environment, interpretation of the misconception tasks data is not possible, as they have not developed any relevant conceptions yet and seem to be making intuitive guesses. Therefore, we developed an additional test part with a very similar format, called the PreProMAT, that will be used as a screening tool to measure relevant, environment-specific basic programming knowledge. This will allow us to interpret the results from the misconception tasks more confidently. The current version of the ProMAT, including the PreProMAT part, can be found on the Open Science Framework (see link above).

Item creation for the PreProMAT part was based on the learning goals stated by current curricula, the specific properties of the programming environment, and from the need to test specific programming commands to be able to assess all relevant misconceptions later. All items in the PreProMAT are also multiple-choice questions, presenting code snippets and one correct answer per item. However, the items are easier than the ones in the ProMAT and each is directed at a simple programming competency, relevant in the learner's programming language. Items for the ProMAT needed to present programmes where specific misconceptions might apply, and some answers for each item were formulated to reflect these misconceptions. All item creation was also guided by available teaching materials (e.g., connected 1 & 2), interactions with a primary school teacher and the programming projects produced by her class, as well as the expertise of two computer scientists that regularly teach programming to teachers in training. Overall, the assessment currently includes 22 tasks measuring the basic programming knowledge (PreProMAT), and 20 tasks testing for possible misconceptions (ProMAT), with an equal number of tasks measuring misconceptions from the four categories described above.

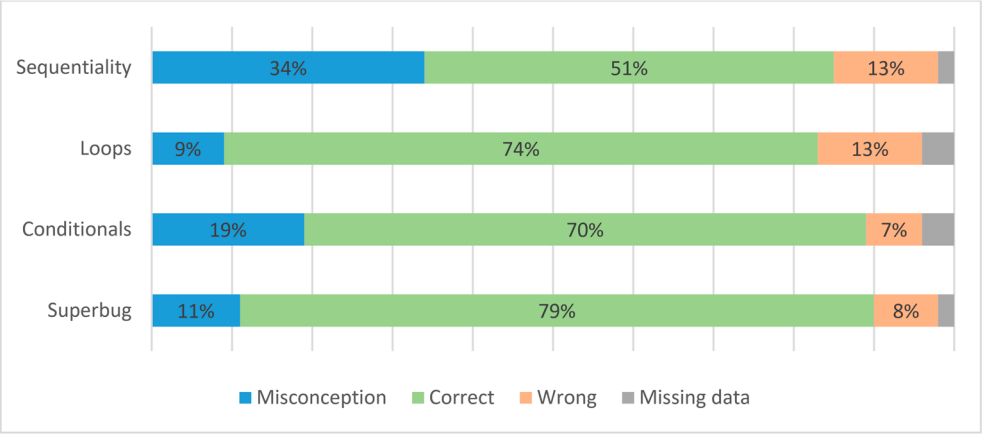
Pilot study

In June and July 2021, we conducted a pilot study with the first version of ProMAT for Scratch. The goals were to eliminate bad items (e.g., too easy, too hard, unclear, lack of item discrimination in psychometric analyses), to see whether specific answers can be linked to specific misconceptions, and to analyse whether our categories were sensible. These analyses are still under way, but a first overview is given here.

Children from one grade 5 classroom (n = 5; 2 girls, 3 boys) and from three grade 6 classrooms (n = 52; 24 girls, 28 boys) participated in this pilot study. One of the teachers was very experienced with Scratch programming and the children had more exposure to programming during class in the weeks leading up to the assessment. All participants were given the first full version of ProMAT and PreProMAT. The five grade 5 children completed the assessment as a think-aloud interview in a one-to-one setting with a researcher. All others completed the assessment for themselves, and of those, six to seven children per classroom (n = 19) were interviewed afterwards. In these interviews, the children were asked about their programming experience, about unclear, difficult, or easy items from the test, and about their reasoning behind some of their answers.

Since Porter et al. (2019) warn that children's performance from think-aloud sessions might be higher on some questions compared to a normal test setting, we excluded the data of the five think-aloud-setting children from the analyses below. Overall, the children were able to complete the assessment - both PreProMAT and ProMAT - within 30 to 40 minutes, with one to two children per classroom not able to complete all tasks within 45 min. On average, the children were able to answer 15.4 out of the 22 PreProMAT items correctly (SD = 3.9) - or about 70%. Before analysing ProMAT data, we excluded 5 children that had very few correct answers (10 or lower) in the PreProMAT. Of the 20 ProMAT tasks, 69% were answered correctly on average (M = 13.7, SD = 4.2), while 19% of answers reflected potential misconceptions (M = 3.7, SD = 2.5). The number of correctly solved tasks in PreProMAT correlated strongly with the number of correct answers in ProMAT ($r(45) = .78$, $p < .001$). Figure 1 shows a breakdown of the average answering patterns between tasks in the different categories.

Figure 1. Results from the ProMAT misconception tasks by category, averaged over all participants. The answers indicate either a possible misconception, the correct answer, a wrong answer that does not reflect a particular misconception, or missing data.



Discussion

The aim of this study was to develop an assessment tool to measure programming misconceptions in children from grades 5 and 6. We outlined this development process and showed initial results from pilot testing, which gave some indication that the ProMAT will be able to measure relevant misconceptions and that the PreProMAT part can screen for sufficient programming experience. Data from PreProMAT was used to exclude data from very low performing children. While we realise that the requirement of some prior experience limits the application of ProMAT as a pre-test, we argue that it is unlikely that data on misconceptions can be confidently interpreted for children that are not familiar enough with the programming environment. The cut-off point chosen here (fewer than half of the PreProMAT tasks correct), likely needs to be adjusted as we gather more data, especially from younger children from grade 5. The strong correlation between correct answers in PreProMAT and ProMAT was to be expected since both tests measure programming ability and operate within the same environment and general task design. This shows that children that were able to answer many basic-knowledge items correctly, also made few mistakes in the misconception items. It also indicates a good internal consistency of the assessment. The proportion of correct answers for the misconception tasks was rather high in the studied population. But since all children were in grade 6 for this analysis and the assessment was conducted at the end of the school year, this result seems acceptable, especially considering that the assessment should also be usable earlier in the school year and for grade 5 children. During the interviews we noticed that the children, especially those that had not worked with Scratch for some weeks, seemed to remember more of the programming mechanisms and improve in performance over the course of the assessment. More detailed analyses are needed to

confirm this assumption. In the tasks from the category Sequentiality children were more likely to choose a 'misconception answer' (34% compared to an average of 19%). While this is an interesting result that we will also look for in the next testing phase, it likely came to be due to a combination of an unfamiliarity of many children in this population with a certain programming command that was used in a few of the Sequentiality-tasks, and a larger number of misconception-specific answer options compared to the other categories. Both aspects will be addressed in the next test version.

Outlook

In the next version of the ProMAT we will change one item containing confusing answer options. In addition, a few misconceptions need to be represented in more of the ProMAT items or be better differentiated. Finally, we will reformat the text and design a bit to improve clarity of presentation. It is also possible that for some test items our participants chose the misconception option for other reasons than for holding the respective misconception. This will become more evident when we further analyse the data from the interviews. This might still lead us to change some of the items or will guide analysis and interpretation of future test results.

In the next steps of the project, a parallel version of ProMAT in xLogo will be developed. After a second pilot testing phase with both versions, we will conduct a larger validation study for the assessment with approximately 500 children in German-speaking Switzerland. This newly validated tool will then be used in an intervention study where we compare the influence of different programming environments on the development of misconceptions. There, we will also use another tool currently in development in this project that assesses more general CS misconceptions. To conclude, we have developed a tool for the assessment of programming misconceptions based on a Swiss primary school curriculum that can be easily implemented within the duration of a school lesson, with first data indicating that children understand the questions well, and that the tool can deliver insights into children's basic programming skills and misconceptions. Further testing will show how well the tool works from a psychometric perspective.

Acknowledgments

We would like to thank all the teachers and children for participating 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

Deutschschweizer Erziehungsdirektoren-Konferenz. (2015). *Schlussbericht der Arbeitsgruppe* zu *Medien und Informatik im Lehrplan 21. 41*(0), 39. https://www.d-edk.ch/sites/default/files/Schlussbericht_MI_2015-02-23_mit_Anhang.pdf

Edelsbrunner, P. A., Schalk, L., Schumacher, R., & Stern, E. (2018). Variable control and conceptual change: A large-scale quantitative study in elementary school. *Learning and Individual Differences, 66*(February), 38–53. https://doi.org/10.1016/j.lindif.2018.02.003

Falkner, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F., McGill, M. M., & Quille, K. (2019). An international comparison of K-12 computer science education intended and enacted curricula. *ACM International Conference Proceeding Series.* https://doi.org/10.1145/3364510.3364517

Grover, S., & Basu, S. (2017). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and Boolean logic. *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE*, 267–272. https://doi.org/10.1145/3017680.3017723

Iten, G. H., Heinz, S., Stöcklin, M., Hübscher, R., & Opwis, K. (2014). The Impact of Interactive Visual Simulations on Learning Statistics. *In CHI'14 Extended Abstracts on Human Factors in Computing Systems*, 2251–2256. https://doi.org/10.1145/2559206.2581208

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: A systematic literature review. In *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*. https://doi.org/10.1145/3293881.3295779

Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies, 23*(4), 1483–1500. https://doi.org/10.1007/s10639-017-9673-3

Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. *Education and Information Technologies, 22*(2), 421–443. https://doi.org/10.1007/s10639-016-9475-z

Pea, R. D. (1986). Language-independent conceptual "bugs" in novice programming To cite this version : *Journal of Educational Computing Research, 2*(1), 25–36. https://doi.org/10.2190/689T-1R2A-X4W4-29J2

Porter, L., Zingaro, D., Liao, S. N., Taylor, C., Webb, K. C., Lee, C., & Clancy, M. (2019). BDSI: A validated concept inventory for basic data structures. *ICER 2019 - Proceedings of the 2019 ACM Conference on International Computing Education Research*, 111–119. https://doi.org/10.1145/3291279.3339404

Rich, K. M., Strickland, C., Binkowski, T. A., Moran, C., & Franklin, D. (2017). K-8 Learning Trajectories Derived from Research Literature. *Proceedings of the 2017 ACM Conference on International Computing Education Research, 9*(1), 182–190. https://doi.org/10.1145/3105726.3106166

Sorva, J. (2008). The same but different: Students' understandings of primitive and object variables. *Koli Calling 2008 - 8th International Conference on Computing Education Research*, 5–15. https://doi.org/10.1145/1595356.1595360

Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education, 13*(2). https://doi.org/10.1145/2483710.2483713

Swidan, A., Hermans, F., & Smit, M. (2018). Programming misconceptions for school students. I*CER 2018 - Proceedings of the 2018 ACM Conference on International Computing Education Research, August*, 151–159. https://doi.org/10.1145/3230977.3230995

Žanko, Ž., Mladenović, M., & Boljat, I. (2019). Misconceptions about variables at the K-12 level. *Education and Information Technologies, 24*(2), 1251–1268. https://doi.org/10.1007/s10639-018-9824-1