Thinking out of the Box

Comparing metaphors for variables in programming education

Hermans, Felienne; Swidan, Alaaeddin; Aivaloglou, Efthimia; Smit, Marileen

DOI
10.1145/3265757.3265765

Publication date
2018

Document Version
Accepted author manuscript

Published in
Proceedings of the 13th Workshop in Primary and Secondary Computing Education, WiPSCE 2018

Thinking out of the Box

Comparing metaphors for variables in programming education

Felienne Hermans*, Alaaeddin Swidan*, Efthimia Aivaloglou+, Marileen Smit*
*Delft University of Technology, the Netherlands
f.f.j.hermans@tudelft.nl,alaaeddin.swidan@tudelft.nl,m.i.e.smit@tudelft.nl
+Open University, the Netherlands
fenia.aivaloglou@ou.nl

ABSTRACT

When teaching novices programming, misconceptions can occur. Misconception are incorrect beliefs about certain programming concept. For example, some novices think that a variable can hold multiple values, in the case of two consecutive assignment statements, such as x = 5; x = 7. While explaining variables introductory materials often use the metaphor of a box for a variable, which might contribute to the 'multiple values' hypothesis. To investigate this, we design and run a controlled experiment with 496 novice programmers, both children and adults. Half of our participants receive an introductory programming lesson in which we explain a variable as a box, while the other half of participants receive the explanation of a variable as being a label. They are subsequently questioned about their understanding of variables. Our results show that, for the simple questions involving one assignment, the box group performs better. However, for questions involving the misconception—with two consecutive assignment statements— the label group outperforms the box group. This however primarily occurs when considering variables of type string, for integers subjects interpret the statements as numeric values to be added.

CCS CONCEPTS

• **Social and professional topics → Computing education**; **Computational thinking**; **K-12 education**;

KEYWORDS

programming education, Scratch, misconceptions

ACM Reference Format:
Felienne Hermans*, Alaaeddin Swidan*, Efthimia Aivaloglou+, Marileen Smit*. 2018. Thinking out of the Box: Comparing metaphors for variables in programming education. In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education (WiPSCE '18), October 4–6, 2018, Potsdam, Germany.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3265757.3265765

WiPSCE '18, October 4–6, 2018, Potsdam, Germany

ACM ISBN 978-1-4503-6588-8/18/10…$15.00
https://doi.org/10.1145/3265757.3265765

1 INTRODUCTION

Teaching programming is notoriously difficult, and attempting to teach novices programming often induces *misconceptions*, i.e. incorrect beliefs about programming concepts. For example, some novice programmers believe that a loop halts as soon as the loop condition is false, rather than first finishing the loop's body. In addition to loops and conditions, one of the concepts that is especially hard to grasp is the concept of a *variable*.

Some novice programmers incorrectly believe that variables can hold multiple values at the same time, or that the variable 'remembers' old values, rather than having the previous value overwritten by a second assignment, while other learners believe that the previous value is still 'somewhere' in the computer even though it is not easily accessible anymore [2, 4, 17, 18, 20]. While the occurrence of misconceptions is a complicated process related to many factors, we hypothesize that one of those factors is the way in which a variable is explained. In particular, while explaining variables, introductory materials often use the metaphor of a box for a variable. They tell learners to envision a variable as a box with a label on it, in which a value is physically stored. This is a nice visual and tangible representation, relating an abstract mathematical concept to an everyday activity like placing something in a box. However, a box could in most cases store not one but two or even more values, especially when those values are represented as little sheets of paper as is often the case.

To investigate the effect of the manner of explaining a variable on the occurrence of the multiple values misconception, we design and run a controlled experiment with 496 novice programmers, both children and adults. Half of our participants receive an introductory programming lesson in which we explain a variable as a box, like a piggy bank or a shoe box. The other half of participants receive an introductory programming lesson in which we explain a variable as being a label that one can place on *one* value, like a temperature or the age of a person. We consistently use the metaphor in both lessons, for example, we use *"x contains 5"* for the box group and *"x is 5"* for the label group. After the programming lesson, the participants receive questions testing their understanding of programming, including both regular questions testing participants' understanding of variables and questions specifically testing the presence of the 'multiple values' misconception. Our results show that for the simple explanation questions, there is no difference between the two groups. However, for questions involving the misconception—with two consecutive assignments— the label group outperformed the box group.

2 RELATED WORK

2.1 Programming education for children

Several researchers have studied teaching novices programming using block-based languages in general, and Scratch in particular. Scratch was taught in middle school classes containing a total of 46 students in the study presented in [12]. Evaluating the internalization of programming concepts, it was found that students had problems with concepts related to initialization, variables and concurrency. In [21], Wilson et al. present an 8-week Scratch course given to 4 primary school classes with a total of 60 students aged 8 to 11, and evaluate it by analyzing the 29 projects that the students created. Maloney et al. [11] taught Scratch as an extracurricular activity, in an after-school clubhouse. By analyzing the 536 students' projects for blocks that relate to programming concepts, they found that within the least utilized ones are boolean operators and variables.

Apart from projects created during courses, other works analyze the public repository of Scratch programs for indications of learning of programming concepts. Yang et al. examined the learning patterns of programmers in terms of block use over their first 50 projects [22]. In [5], the use of programming concepts was examined in relation to the level of participation, the gender, and the account age of 5 thousand Scratch programmers. Moreno and Robles analyzed 100 Scratch projects to detect bad programming habits related to Naming and Duplication [13]. In prior work, we have analyzed 250 thousand Scratch projects in terms of complexity, used programming concepts and smells [1]. Seiter and Foreman [16] proposed a model for assessing computational thinking in primary school students and applied it on 150 Scratch projects, finding that design patterns requiring understanding of parallelization, conditionals and, especially, variables were under-represented by all grades apart from 5 an 6.

2.2 Programming misconceptions

A programming misconception is an incorrect understanding of a concept or a set of concepts, which leads to making mistakes in writing or reading programs [19]. Misconceptions can be related to all sorts of programming concepts, not just advanced ones. Often even they are related to language-independent basic constructs like looks, variables or control flow, which, although simple to experienced programmers are particularly difficult for novices to learn.

One example of a common programming misconception is the one that this paper examines, the belief that a variable can hold multiple values. This means a learner thinks that this code snippet above leads to temperature being 5 *and* 7:

```
temperature = 5;
temperature = 7
```

Another common misconception is that variable assignment works in both directions. That means programming novices believe that `temperature = 5;` means the same as `5 = temperature`. Assumptions about this particular misconception are that it stems from mathematics education, where children are taught that `1 + 2 = 3` and also `3 = 1 + 2`, meaning that both sides of the equation = may be interchanged.

Early efforts researching misconceptions include studies in Pascal [17], BASIC [14] and Prolog [6]. Further studies followed for OO languages such as Java [7]. Common misconceptions in Java are related to the scope of variables, modularization and decomposition, and inheritance [7, 9].

Research has also focused on understanding origins of misconception. A programming misconception does not mean the student has a complete lack of knowledge, rather they have some knowledge but miss the full picture. Often, some knowledge comes from related domain like natural language of math. However, du Boulay [2] introduced what he called the 'notional machine' as an origin of programming misconceptions. The notional machine refers to the general properties that a student assumes of the machine executing their code. It involves various aspects related to the program: compiler, memory management, etc. Having an incorrect understanding of the notional machine of a programming language is believed to be the cause of many misconceptions [10, 19]. For example errors were found as a result of the students assuming that 'there is a hidden, intelligent mind within the computer that helps the programmer to achieve their goals', or 'forgetting about alternative branches because they are too obvious to merit consideration' or that or that two variables may not refer to the same object [8].

Finally, the variety of misconceptions make it difficult for educators to take full account of. In this regard, Sorva [20] provides a comprehensive summary of programming misconceptions reported by various researchers [2, 4, 9, 14, 17, 19]

3 EXPERIMENTAL SETUP

A common misconception related to variables that novice programmers have is that a variable can hold multiple values or that it 'remembers' previous values. The goal of our study is to understand what the impact is of the metaphor that is used to introduce a variable as programming concept. More specifically, we compare the metaphor of a variable as a box that holds a value, to the metaphor that a variable is a value with a label. With the study we answer the following research questions:

- R_1 Does the metaphor of explaining variables increase the understanding of the concept of a variable in general?
- R_2 Does the metaphor of explaining variables decrease the likelihood that a participants develops the 'multiple values misconception'?

Associated with these research questions are two null hypotheses, which we formulate as follows:

- $H1_0$ The metaphor of explaining variables does not impact the understanding of the concept of a variable in general.
- $H2_0$ The metaphor of explaining variables does not impact the likelihood that a participants develops the 'multiple values misconception'.

To test these null hypotheses, we create two lessons that introduce a variable, one using the box metaphor and one using the label metaphor, and both groups receive a test of variable knowledge after the lesson. We use a between-subjects design, meaning every participant is either in the box or in the label group.

3.1 Participants

In total, 496 people participated in our experiment, 322 children and 174 parents, see Figure 3. In total our experiment had 253 female participants (mothers and girls) and 235 male participants (fathers and boys). 8 participants entered no gender. We only recorded age for the 332 children in the experiment, of which 8 children did not enter it. The ages of the remaining 324 children are shown in Figure 6.

3.2 Setup

We ran this experiment in the NEMO science museum in Amsterdam, as part of the Science Live project where scientists can run experiments in the museum[1]. Visitors of the museum were asked to join in an experiment on programming but received no further information on what the experiment would measure. We however did tell them that they might get a different lesson than the one their parents or siblings would receive. Participants did not get financial compensation for participation, but children participating did receive a certificate for their efforts. The experiment was conducted in a separate room in the museum that seated 8 people at a time. In total we spent 14 days at the museum, running the experiment for about 5 hours each day.

3.3 Lessons

As explained above, the goal of this paper is to explore the effect of metaphors used in explaining variables to novice programmers. We therefore designed two different introductory programming lessons explaining the concept of a variable to novice programmers, using the programming language Scratch. Scratch is a block-based programming language. Scratch is a block-based programming language aimed at children, developed by MIT. Scratch can be used to create games and interactive animations, and is available both as a stand-alone application and as a web application. Figure 1 shows the Scratch user interface in the Chrome browser. For an extensive overview of Scratch see [3, 15].

We separated the subjects into two random groups: 244 of the participants received a lesson using the metaphor of a box for a variable, while 252 received a lesson using the label metaphor. [2] We assumed no previous programming knowledge. To the 'box' group, we explained a variable as being a box, like a piggy bank or a shoe box in which you can store a value. The 'label' group received in which we explain a variable as being a label that one can place on *one* value, like a temperature or the age of a person. We consistently use the metaphor in both the box and the label lessons, For example, we use *"x contains 5"* for the box group and *"x is 5"* for the label group.

3.4 Language

We offered the lesson and the test in both Dutch and English, since the science museum is regularly visited not only by Dutch families but also by tourists. Still the majority of participants (341) chose to receive the lesson and the test in Dutch, as seen in Figure 4, while the remaining 155 participants chose English. In some cases, parents translated the lesson or the questions for their children, who spoke neither of the two languages well enough.

[1] https://www.sciencelive.nl/

[2] All materials can be found online at http://www.felienne.com/archives/6063

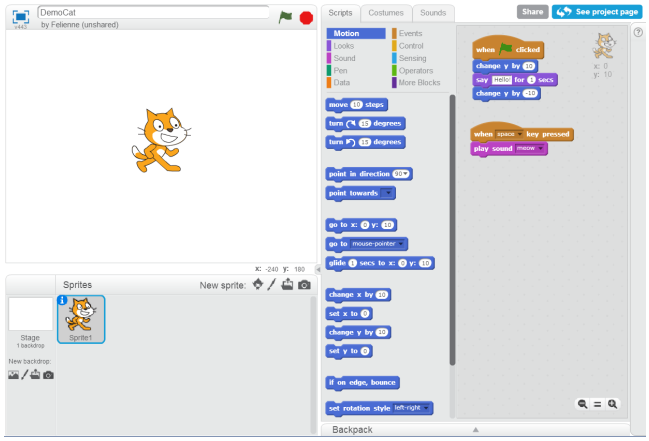

Figure 1: The Scratch user interface consisting of the 'cat' sprite on the left, the toolbox with available blocks in the category 'motion' in the middle and the code associated with the sprite on the right. The upper right corner shows the actual location of the sprite.

3.5 Test

After the programming lesson using one of the two metaphors, the participants received questions testing their understanding of programming, including both regular questions testing the participants's understanding of variables, and questions specifically testing the presence of the 'multiple values' misconception. Table 1 lists the questions we use in the test. Figure 2 shows a small excerpt from the English, box group test. The label groups differs in the formulation of questions slightly, using *"what is name?"* rather than *"what is stored in name?"*.

The test contains questions that test whether participants can correctly predict the outcome of a given piece of code including variables, combined with open text questions with which we attempt to understand the thinking of participants deeper. We take both answers into account when grading answers as correct or incorrect, since sometimes incorrect reasoning can lead to a correct answer. For example, some participants think they need to add 0 and 2 to calculate the correct answer to question 7 (in Figure 2). While this is the correct answer, it is not the correct reasoning.

4 RESULTS

4.1 R_1: Impact of the metaphor on general variable understanding

R_1: Does the metaphor of explaining variables increase the understanding of the concept of a variable in general? Table 2 shows the percentage of correct answers to all tracing questions in the test. Where differences are significant, as calculated by a Chi-Square test, the z score and p value are provided. Questions 1, 2, 3, 5, 6 and 12 concern basic understanding of variables, since in those questions only one assignment is present.

Table 1: Questions in the final test (English version, box version)

Question ID	Data Type	Type	Category	Question
1	Integers	Tracing	General understanding of variables	We use this code: [set points to 2] How much is points?
2	Integers	Tracing	General understanding of variables	We use this code: [set points to 2] How much is points + 5?
3	Integers	Tracing	General understanding of variables	Is points 7?
4	Integers	Open Text	General understanding of variables	Why did you choose that answer?
5	Integers	Tracing	General understanding of variables	We use these blocks: [set points to 4] [change points by 2] What is stored in points now?
6	Integers	Tracing	General understanding of variables	We use these blocks: [set points to 7] [change points by -1] What is stored in points now?
7	Integers	Tracing	Multiple values misconception	We use these blocks: [set points to 0] [set points to 2] What is stored in points now?
8	Integers	Open Text	Multiple values misconception	Why did you choose that answer?
9	Integers	Tracing	Multiple values misconception	We use these blocks: [set a to 10] [set b to 20] [set a to b] What is stored in a now?
10	Integers	Tracing	Multiple values misconception	We use these blocks: [set a to 10] [set b to 20] [set a to b] What is stored in b now?
11	Integers	Open Text	Multiple values misconception	Why did you choose that answer?
12	Strings	Tracing	General understanding of variables	We use this code: [set name to John] What is stored in name now?
13	Strings	Tracing	Multiple values misconception	We use these blocks, what is stored in name? [set name to Karl] [set name to Hassan] What is stored in name now?
14	Strings	Open Text	Multiple values misconception	Why did you choose that answer?

As can be seen from the table, participants in the box group perform somewhat better on all of these questions, apart from question 12, where the data type is string. A Chi-Square Test shows that this difference is significant for questions 2 (z = 2.128, p = 0.033), 3 (z = 2.506, p = 0.012) and 5 (z = 2.12, p = 0.034). This means we reject $H1_0$, and conclude that the metaphor by which we explain variables impacts the basic understanding of the concept of a variable in general, and that the box metaphor *strengthens* this understanding.

The box metaphor increases participants understanding of the basic working of variables.

4.2 R_2: Impact of the metaphor on the two values misconception

R_2: Does the metaphor of explaining variables decrease the likelihood that participants develop the 'multiple values misconception'?

To answer this research question, we analyze the answers to the questions with multiple assignments: Questions 7, 9 and 10 about integers and 12 and 13 about strings. Out of these questions, there is a significant difference only for Question 9 in favor of the label group, if we analyze the correctness of answers.

However, we are interested in more than simply the ability of participants to answer the question correctly. We also explore the presence of the misconception that a variable can hold multiple

Table 2: Correctness of answers to the tracing questions, for the box and the label groups respectively. For signification differences, the best group, and the z score and p value are provided.

Question ID	Box Correct	Label Correct	Significant?	Best group	z-score	p value
1	80.7%	74.2%	No	—	—	—
2	77.9%	69.4%	Yes	Box	z = 2.128	p = 0.033
3	84.4%	74.4%	Yes	Box	z = 2.506	p = 0.012
5	80.3%	72.2%	Yes	Box	z = 2.12	p = 0.034
6	84%	79%	No	—	—	—
7	79.1%	75.8%	No	—	—	—
9	16%	26.6%	Yes	Label	z = 2.88	p = 0.004
10	61.5%	54/8%	No	—	—	—
12	76.6%	80.6%	No	—	—	—
13	20.9%	27.8 %	No	—	—	—

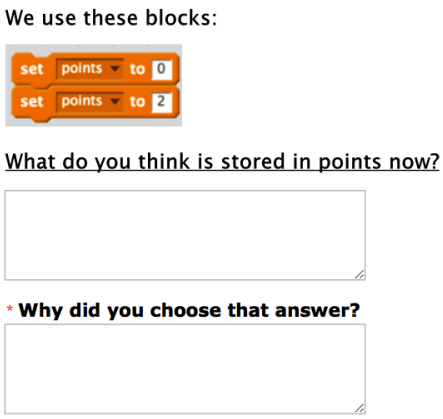

Figure 2: Questions 7 and 8 as presented to the participants (English version, box version)

Figure 3: Participants in our study, divided over the two lessons.

Figure 4: Language in which the participants were presented the lesson and the subsequent test

Figure 5: Gender of all participants in our study

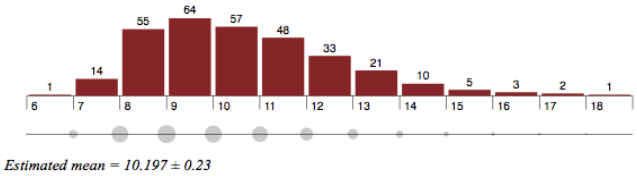

Figure 6: Ages of children in our study

values. Therefore, we code the answers to Questions 7, 9 and 10, and 13 plus their corresponding open text questions, Questions 8, 11 and 14 and analyze which (mis)conceptions participants have about the number of values a variable can hold. Here we do not take into account whether they have the correct value in case they believe the one value hypothesis, but focus on the fact that those participants at least understand that a variable holds one value at a time. Sometimes the (mis)conception can be seen directly from the answers to the Tracing questions, for example, some participants answer "KarlHassan" to question 13, or some even tried "KarlSan", representing an interesting variation of the multiple values hypothesis. Sometimes however, the misconception can be found in the open questions.

When analyzing the answers to Questions 7, 9 and 10, where participants were tracing, combined with the open text answers to Questions 8 and 11, we surprisingly found hardly any occurrences of the multiple values misconception, but we encountered a different misconception. Many participants interpreted the question as an addition, stating they needed to add the two values to get the right answer, leading to 2 for Question 7 and 30 for 9 and 10. For Question 7, accidentally, this is indeed the correct, so there we only marked 2 as correct with a correct explanation as answer to Question 8. Some participants had a slightly different misconception, subtracting the value 20 from 10 resulting in the answer −10. We classified that answer as the addition misconception too.

We suspect that participants were applying a pattern here, either from earlier questions (Questions 2, 5 and 6) where indeed a value was added to a variable, or they were more broadly applying addition as a pattern they know leads to correct answers in assignments as are they are common in school. As one of the participants said, *"these are like sums in arithmetic class"*. This misconception seems related to misconception 15 from [20]: 'Primitive assignment

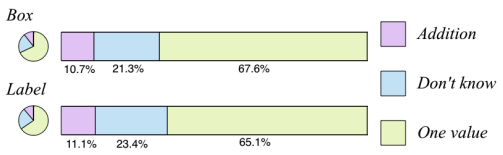

Figure 10: Classification of answers to Question 10. No difference between the two groups.

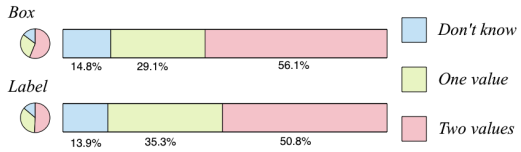

Figure 11: Classification of answers to Question 13. Box group is more likely to suffer from multiple value hypothesis.

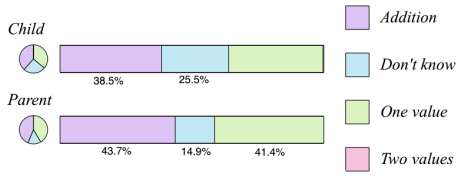

Figure 7: Answers to Question 13. No significant difference between parents and children.

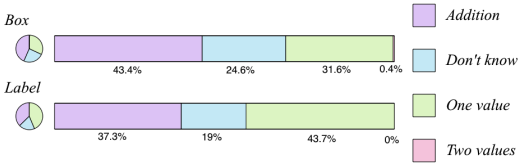

Figure 8: Classification of answers to Question 7. Box group is more likely to believe the addition misconception and less likely to believe one value hypothesis.

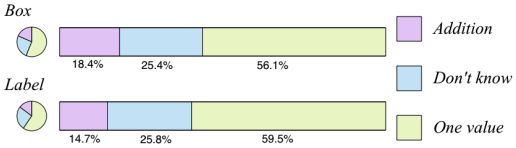

Figure 9: Classification of answers to Question 9. No difference between the two groups.

stores equations or unresolved expressions' [2, 4, 14, 19]. We first suspected that this misconception was due to children applying lessons from mathematics in school, but we found both in parents and in children alike, as demonstrated by Figure 7.

The coding of the answers to Questions 7 and the corresponding open text question 8 can be found in Figure 8, for the box and the label group, and Figure 9 and 10 show the classification of the answers of Questions 9 and 10 and the corresponding open text question, Question 11. For Questions 7 and 9, we find that the label group is more likely to hold the (correct) one value hypothesis than the box group. For Questions 9 and 10 there is no difference.

In the answers to Question 13/14, where the data type of the variable in the question is a string, the multiple values hypothesis does occur more often in both groups, but significantly more in the box group (Chi-Square: z = 2.526, p=0.012). This means we reject $H2_0$, and conclude that the metaphor by which we explain variables impacts the probability that a novice programmer develops the multiple values misconception, and that the box metaphor *increases* this chance.

> Novice programmers in Scratch are more likely to suffer from the misconception that values are added up when there are multiple assignments in code.
> For strings, the box metaphor increases the probability that a participant holds the multiple values misconception.

5 DISCUSSION

5.1 A new misconception

While examining the effect of boxes and labels as metaphors for variables, we have uncovered a new misconception: novice programmers apply patterns like calculation to programming problems. As far as we know such a misconception was not detected in university level computer science students. We hypothesize this is due to the fact that they are well aware that programming is not so similar to basic mathematics that pattern from mathematics could be the solution.

5.2 Impact of metaphors

The fact that we measure differences in performance and conception after a short programming lesson indicates that programming educators should use metaphors with care. While the box metaphor supports initial understanding, it is prone to confuse novice programmers when programs get more complicated. What the implications of this are precisely is unclear. Should we abolish the box metaphor entirely, knowing that this makes learning about variables harder? Or should we use it, but later explain to learners that this metaphor was flawed and refine it? These are open questions that we will study in future experiments.

5.3 Impact of the datatype

One of the surprising findings of this study is that there is a clear difference between the answers given for questions about integers as datatypes(Q1 to Q11) and those about strings (Q12 and Q13). Participants are more likely to develop the multiple values misconception with strings than with integer values! This was not what we expected, and testing these differences was not part of our original set of research questions. One of the reasons we imagine for this could be that participants are more confused by string variables, because they are not used to calculate with strings, while they are used to calculating (integer) numbers. Examining the open answers

participants gave, especially to Q8, we observed that participants often tried to calculate with the given values in Q7, i.e. adding 0 and 2 together, rather than understanding that 2 overwrites the existing value 0. Similarly in Questions 9 and 10 people answered 30 as an answer.

This could be caused by Question 2 in which participants indeed needed to add values. With that question we wanted to test whether the subjects understood that a variable is something you can work with, but in retrospect maybe it was adding to the confusion. This requires more study, for example with a new questionnaire where questions about the multiple values misconception are not preceded by calculation questions.

5.4 Threats to validity

There are a number of threats to the internal validity of this study. Firstly some subjects might have been aware of the goals of the study, since they tried to compare their lesson to that of their siblings or parents. We mitigated this by separating family members away from each other, but we did not always succeed in preventing them from talking about the questions. We however minimized this threat by not revealing to the participants what type of experiment we were performing, and told them there were different questions to prevent them from cheating.

There are threats to the external validity of our study too. The generalizability of our results could be impacted by both limited representativeness of the simple assignments, and the participating subjects. Although our sample size of almost 500 participants is large, people visiting a science museum and willing to participate in a programming study of course do not represent the programming interest of the general public.

6 CONCLUDING REMARKS

The aim of this paper is to examine the effect of the metaphor used when explaining the concept of a variable to novice programmers in the context of block-based programming language Scratch. As such we have firstly designed two introductory programming lessons, one using the box and the other using the label metaphor. We subsequently evaluated those lessons in a controlled experiment with 496 children and their parents. The results of this evaluation show that subjects that have follow the box lesson demonstrate a better understanding of variables used in simple programs, but are more likely to suffer from the misconception that a variable can hold multiple values when reading programs with multiple assignment statements. In addition to these results, we uncover a new misconception for novice programmers: they interpret programming puzzles as mathematics exercises. The main contributions of this paper are the design (Section 3) and execution (Section 4) of a controlled experiment into the differences between the box and the label metaphor for explaining variables to novice programmers.

The current work gives rise to several avenues for future work. Firstly, of course, bigger and more extensive experiments with a more diverse range of subjects are needed. Would we measure a different effect on participants with some programming experience? Can the two value misconception be easy resolved after it has been used successfully as an introductory tool?

Furthermore, we plan to explore a more diverse range of misconceptions in a similar fashion. These could be misconceptions related to variables, like the fact that the value of a variable is related in some fashion to the natural-language semantics of its name [2, 14, 17], or that two variables may not refer to the same object. Are those effected by the box or label metaphor too? We could even extent this to other misconceptions, like the misconception that a loop ends as soon as the loop condition [2] becomes false could be explained with and without references to the word and concept of *while* in natural language.

REFERENCES

[1] Efthimia Aivaloglou and Felienne Hermans. 2016. How Kids Code and How We Know: An Exploratory Study on the Scratch Repository. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. ACM, 53–61. https://doi.org/10.1145/2960310.2960325

[2] Benedict Du Boulay. 1986. Some Difficulties of Learning to Program. *Journal of Educational Computing Research* 2, 1 (1986), 57–73. https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9 arXiv:https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9

[3] Karen Brennan, Christian Balch, and Michelle Chung. 2014. *CREATIVE COMPUTING*. Harvard Graduate School of Education.

[4] D. Doukakis, M. Grigoriadou, and G Tsaganou. 2007. Understanding the Programming Variable Concept with Animated Interactive Analogies. *Proceedings of the The 8th Hellenic European Research on Computer Mathematics & its Applications Conference, HERCMA '07* (2007).

[5] Deborah A. Fields, Michael Giang, and Yasmin Kafai. 2014. Programming in the Wild: Trends in Youth Computational Participation in the Online Scratch Community. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14)*. ACM, 2–11. https://doi.org/10.1145/2670757.2670768

[6] Pat Fung, Mike Brayshaw, Benedict Du Boulay, and Mark Elsom-Cook. 1990. Towards a taxonomy of novices' misconceptions of the Prolog interpreter. *Instructional Science* 19, 4-5 (1990), 311–336. https://doi.org/10.1007/bf00116443

[7] Ken Goldman, Paul Gross, Cinda Heeren, Geoffrey Herman, Lisa Kaczmarczyk, Michael C. Loui, and Craig Zilles. 2008. Identifying important and difficult concepts in introductory computing courses using a delphi process. *ACM SIGCSE Bulletin* 40, 1 (2008), 256. https://doi.org/10.1145/1352322.1352226

[8] Simon Holland, Robert Griffiths, and Mark Woodman. 1997. Avoiding Object Misconceptions. *SIGCSE Bull.* 29, 1 (March 1997), 131–134. https://doi.org/10.1145/268085.268132

[9] Linxiao Ma. 2007. *Investigating and improving novice programmersâ€™ mental models of programming concepts*. Ph.D. Dissertation. University of Strathclyde, Glasgow, UK. http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.444415

[10] L. Ma, J. Ferguson, M. Roper, and M. Wood. 2011. Investigating and improving the models of programming concepts held by novice programmers. *Computer Science Education* 21, 1 (2011), 57–80. https://doi.org/10.1080/08993408.2011.554722

[11] John H. Maloney, Kylie Peppler, Yasmin Kafai, Mitchel Resnick, and Natalie Rusk. 2008. Programming by Choice: Urban Youth Learning Programming with Scratch. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '08)*. ACM, 367–371. https://doi.org/10.1145/1352135.1352260

[12] Orni Meerbaum-Salant, Michal Armoni, and Mordechai (Moti) Ben-Ari. 2010. Learning Computer Science Concepts with Scratch. In *Proceedings of the Sixth International Workshop on Computing Education Research (ICER '10)*. ACM, New York, NY, USA, 69–76. https://doi.org/10.1145/1839594.1839607

[13] J. Moreno and G. Robles. 2014. Automatic detection of bad programming habits in scratch: A preliminary study. In *2014 IEEE Frontiers in Education Conference (FIE)*. 1–4. https://doi.org/10.1109/FIE.2014.7044055

[14] Ralph T. Putnam, D. Sleeman, Juliet A. Baxter, and Laiani K. Kuspa. 1986. A Summary of Misconceptions of High School Basic Programmers. *Journal of Educational Computing Research* 2, 4 (1986), 459–472. https://doi.org/10.2190/fgn9-dj2f-86v8-3fau

[15] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for All. *Commun. ACM* 52, 11 (Nov. 2009), 60–67. https://doi.org/10.1145/1592761.1592779

[16] Linda Seiter and Brendan Foreman. 2013. Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*. ACM, 59–66. https://doi.org/10.1145/2493394.2493403

[17] D. Sleeman, Ralph T. Putnam, Juliet Baxter, and Laiani Kuspa. 1986. Pascal and High School Students: A Study of Errors. *Journal of Educational Computing Research* 2, 1 (1986), 5–23. https://doi.org/10.2190/2XPP-LTYH-98NQ-BU77 arXiv:https://doi.org/10.2190/2XPP-LTYH-98NQ-BU77

[18] Elliot Soloway, Kate Ehrlich, Jeffrey Bonar, and Judith Greenspan. 1982. What do novices know about programming. *Directions in human-computer interaction* (1982), 87–122.

[19] Juha Sorva. 2008. The same but different students' understandings of primitive and object variables. *Proceedings of the 8th International Conference on Computing Education Research - Koli '08* (2008). https://doi.org/10.1145/1595356.1595360

[20] Juha Sorva. 2012. Visual program simulation in introductory programming education; Visuaalinen ohjelmasimulaatio ohjelmoinnin alkeisopetuksessa. (2012), 428 pages. http://urn.fi/URN:ISBN:978-952-60-4626-6

[21] Amanda Wilson, Thomas Hainey, and Thomas Connolly. 2012. Evaluation of computer games developed by primary school children to gauge understanding of programming concepts. In *European Conference on Games Based Learning*. Academic Conferences International Limited, 549.

[22] Seungwon Yang, Carlotta Domeniconi, Matt Revelle, Mack Sweeney, Ben U. Gelman, Chris Beckley, and Aditya Johri. 2015. Uncovering Trajectories of Informal Learning in Large Online Communities of Creators. In *Proceedings of the Second ACM Conference on Learning @ Scale*. ACM, 131–140. https://doi.org/10.1145/2724660.2724674