Different instruction methods for learning programming concepts in Scratch

Testing the retention of programming misconceptions among primary school children

Susanne Spek, Iris Yocarini and Felienne Hermans
Media Technology MSc program, Leiden University
Leiden, The Netherlands
s.spek.2@umail.leidenuniv.nl
July 2020

Abstract

Programming education is an important research topic but is mostly focused on students. Nowadays, there is more interest in giving programming lessons to young children, but still less programming research is focused on this sample group. Learning programming is a difficult process and it is almost unavoidable that misconceptions arise. A misconception is an incorrect understanding of a concept. Because misconceptions negatively affect performance it is important to avoid misconceptions while learning how to program. To avoid misconceptions, someone must have a correct understanding of the concept, which may be influenced by the used instruction method. In this study the effect of three different instruction methods, on the retention of programming misconceptions among 10- to 12-year old children is tested. The used instruction methods are: worked examples, given subgoal labelling and self-explaining subgoal labelling. Nineteen children of primary school took part in this study. Each participant followed three online programming classes in Scratch where the basic programming concepts, variables, loops, if/else statements, and input were learned. Misconceptions were measured twice, during and after the intervention, by using a multiple-choice questionnaire with programming exercises in Scratch. No significant differences were found between the number of misconceptions and used instruction method over time. So, on average no difference in the effect of misconceptions over time for different instruction methods is found. Analyzing the descriptive statistics suggests the subgoal labelling instruction methods to be more effective for learning programming concepts. In those conditions there were fewer misconceptions, less wrong answers and more correct answers compared to the worked example condition. This pilot study showed insights and directions for further research. Further research must show whether a specific instruction method is more effective in learning programming concepts.

1. INTRODUCTION

Programming is a hot topic and the importance of learning programming from an early age is being recognized more and more. If children start with learning the basic programming concepts at a young age, they benefit from this during programming classes in higher education. Children who learned Scratch benefited from this in secondary programming classes, they needed less time to learn a new topic, had fewer difficulties with learning and they achieved higher cognitive levels of understanding the concepts (Armoni, Meerbaum-Salant, & Ben-Ari, 2015). Therefore, learning programming at a young age can be beneficial for programming classes in higher education where the high drop-out rate is seen as a problem (Vihavainen, Airaksinen, & Watson, 2014). Learning programming at a young age can be stimulated by programming classes in primary schools. In some countries, they already integrated it in the primary school's curriculum (Hermans & Aivaloglou, 2017).

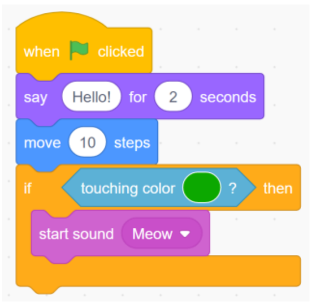
However, programming education brings some difficulties since programming is hard to learn and understand for novices. Many people face difficulties at the beginning when learning to program (Gomes & Mendes, 2007). Therefore, it is important to give programming classes to young children in a way that is beneficial for the learner. The way programming classes are given, the instructional approach, may influence the understanding of programming concepts and plays a major role in the learning process. However, the programming community lacks a collective memory of how programming classes should be given (Hermans & Smit, 2018). That is why this research focussed on the effect of different instructional approaches in programming education with 10- to 12-year old primary school children.

An element increasing the difficulty to learn how to program is the existence of many concepts in programming, such as statements and loops. In the process of learning those programming concepts, misconceptions may occur. A misconception is an incorrect understanding of a concept, while having confidence that it is correct. Misconceptions affect performance (Simon, 2011) and can lead to making mistakes in writing or reading computer programs (Sorva, 2008). They can remain for a long time (Simon, 2011). Therefore, it is important to avoid misconceptions while learning to program.

Having a misconception is not always about a lack of knowledge (Du Boulay, 1986) but rather about having the incorrect understanding of the knowledge. When people have a misconception, they think they have the correct understanding of the concept. Misconceptions can also be caused by self-interpreted knowledge from other domains (Du Boulay, 1986). To avoid misconceptions, someone must have a correct understanding of the concept, in other words; someone must have a correct conceptual knowledge. This may be influenced by the used instruction method. In this study the relation between instruction method and number of misconceptions was explored.

Misconceptions remain for a long time and therefore, it is important to test them over a longer period of time. In this study, the one-week retention of programming misconceptions about variables, loops, if/else statements and input is tested among 10- to 12-year old children.

In this study, Scratch was used to teach 10- to 12-year old primary school children programming concepts. The programming misconceptions are measured with a questionnaire with exercises in Scratch. Scratch is an example of a visual programming language which is primarily used by young people and primary schools to learn the basics of programming. See Figure 1 for an example of a Scratch program. It has been shown that it is possible to learn programming concepts in Scratch (Meerbaum-Salant, Armoni, & Ben-Ari, 2013).



```
when [flag] clicked
say Hello! for 2 seconds
move 10 steps
if touching color [ ] ? then
    start sound Meow
```

Figure 1: An example of Scratch

Although a lot of research has been done concerning instruction methods in primary schools, this is not done much in programming education for children. In mathematics there was found that problem-based learning is more effective in improving mathematical literacy compared to direct instruction (Firdaus, 2017) and in research about science education in primary schools it was found that children learned more from direct instruction (Klahr, D., & Nigam, M. (2004). Programming education differs from regular subjects in education because of the possibility of exploring concepts and the relentless of the programming language (Hermans & Smit, 2018). All concepts in a programming environment are available to use from the start and no mistake is disregarded. Therefore, researching programming education in depth can be relevant. Whereas research has been conducted on instruction methods of programming education, these studies mainly used student samples and little work has focused on primary school children. The studies that used children as sample, mainly did not focus on the knowledge or retention of programming concepts but rather on the performance of children during class. The effectiveness of different instruction methods among children during class is for example measured with completed levels (Joentasuta & Hellas, 2018) or task success during class (Ichinco, Harms & Kelleher, 2017).

This study focused on the relation between the retention of programming misconceptions among primary school children and instruction method. The programming concepts were learned with the use of Scratch. The aim of the study was to answer the following research question.

RQ *Is there a relation between the instruction method used in programming education and the one-week retention of programming misconceptions among primary school children who learn Scratch?*

2. BACKGROUND

2.1 Programming misconceptions

There is a broad variety of misconceptions and misconceptions can occur for all programming concepts, also the basic ones (Swidan, Hermans, & Smit, 2018) such as variables and loops. Sorva (2012) classified the misconceptions known from literature in a comprehensive list of 162 misconceptions. It is impossible to take them all into account and not each misconception occurs in all programming languages. Also, misconceptions are based on level of expertise. For example, a novice programmer who only learned variables and loops so far cannot have any misconception about objects because he or she has no knowledge about the concepts of an object. This study focused on misconceptions that are relevant while learning Scratch. The most difficult concepts to learn are variables, loops, and conditional statements (Du Boulay, 1986). Research from Swidan, Hermans & Smit (2018) concluded the three most common programming misconceptions in Scratch. These top three misconceptions in Scratch are; the difficulty of understanding the sequentially of statements, the difficulty of understanding that a variable holds one value at a time and the difficulty to understand the interactive nature of a program when user input is required.

2.2 Instruction method

2.2.1 Worked examples

Education can be designed and given with many different instructional approaches. Additionally, there are many conflicting results on the effectiveness of different instruction methods. Where some papers of research promote a certain instruction method, others disagree with this. One of the instruction methods with conflicting results on effectiveness is the worked example. Worked examples provide people with a step-by-step guideline to solve the problem (Atikinson, Derry, Renkl & Wortham, 2000; Sweller & Cooper, 1985). This method is a good way of learning for people without experience (Cooper & Sweller, 1987; Kalyuga, Chandler, Tuovinen & Sweller, 2001). It helps with learning cognitive skills and develop knowledge structures, which can be important at the beginning of learning problem-solving processes (Atkinson, Derry, Renkl, & Wortham, 2000). People learn more efficiently with the use of a worked example (Mclaren et al, 2014). While some studies claim that worked examples are effective, some study results show problems with using this method. They would not help in understanding conceptual knowledge (Catrambone, 1998) and it can lead to ineffective recall and transfer (Bransford, Brown, & Cocking, 2000). This might be because people memorize the steps instead of learning how to distinguish the key features from the worked example and learn how to solve the problem by conceptual understanding (Catrambone, 1998).

Research has shown that worked examples may be effective in programming education (Joentausta & Hellas 2018; Rahman & Du Boulay, 2010). Students who had access to a worked example to solve a programming task completed more tasks within a fixed time, this suggests that worked examples may improve the learning efficiency for students when programming (Zhi, et al., 2019). Also, in programming education some problems with using worked examples to learn programming were found. For novices it can be difficult to use the worked example effectively. It can be hard to understand the examples and to find relevant blocks to finish a programming task (Ichinco, Harms and Kelleher, 2017). This study evaluates the worked example in programming education for children.

2.2.2 Subgoal labelling

Subgoal labelling is a technique where the steps from the worked example are grouped in subgoals and labelled with meaningful names. Subgoal labelling can help children learn how specific subgoals are solved. This technique ensures learners to focus on higher-level features of the problem (Atkinson, Derry, Renkl, & Wortham, 2000). During programming, it is important to think in steps and subgoals which is sometimes hard for children. Children may benefit from subgoal labels while learning programming (Joentausta & Hellas 2018), because it helps to understand the subgoals of the problem-solving process (Catrambone, 1950). Joentausta and Hellas (2018) found that children who received worked examples with subgoal labels completed more levels in Lightbot (a programming game) than children who only received the worked example. However, this research only tested the completed levels during one class and did not focus on retention or conceptual knowledge. This is important for this study since it has a focus on misconceptions which are related to conceptual knowledge. Research from Margulieux and Catrambone (2016) tested on conceptual knowledge in programming education but focused on students. They suggested that subgoal labelling can promote retention and transfer in a programming task and that it can improve problem-solving performances.

2.2.3 Constructive learning

Worked examples and subgoal labelling are both active learning techniques. When using an active learning type, the learners engage with the instructional materials with some form of motoric action or physical manipulation (Chi & Wylie, 2014). In both instruction methods children repeat and follow the given instructions which are forms of action. Research suggests that a constructive learning technique can be more effective (Chi, 2009). In a constructive learning type, the learner generates additional knowledge beyond the provided instructions and given information (Chi & Wylie, 2014). That is why a third instruction method is compared during this research. Subgoal labelling is not only used as an active learning type where the subgoals are already given but, is also used as a constructive learning type by letting the participants give a name to each subgoal. With this method, participants are asked to explain and be actively busy with the material which may results in them being more involved and aware of what they are doing. During the worked example and subgoal labelling with given names, they can follow the steps without thinking about the relevance of every step. On the other hand, giving a name to a label can be too difficult for children and can distract of the learning process. Subgoal labelling where people must name the subgoals themselves has so far only been tested with students. This method is effective for students if they get some instructional support, such as hints or feedback while constructing the labels (Margulieux and Catrambone, 2016).

2.3 Scratch

To make programming more attractive and easier to learn for children different programming languages such as LightBot, LOGO and Scratch have been developed. Those programming languages make use of visual blocks to make programming user-friendly for children and to ensure they focus on the programming concepts instead of syntax. Consequently, children using Scratch can master programming concepts more quickly (Armoni, Meerbaum-Salant, & Ben-Ari, 2015; Price & Barnes, 2015) and no syntax errors can be made. Research has also shown that children have fewer misconceptions when using a visual programming language compared to text-based programming languages (Mladenovic, Boljat, & Zanko, 2018).

For programming, a high level of abstract thinking is necessary, which not all children master. Children with an average problem-solving skill perform better in Scratch, compared to text-based programming languages. This is because Scratch helps children to develop their problem-solving skills (Meerbaum-Salant, Armoni, & Ben-Ari, 2013).

The use of Scratch is positive for decreasing the number of misconceptions because Scratch highly motivates students (Ouahbi et al, 2014). Children with a positive attitude about the material are more motivated to learn and often show an active learning attitude, which results in better learning performance (Castejón, Gilar & Pérez, 2006; Schunk & Zimmerman, 2012)

3. METHODOLOGY

The goal of the study was to measure programming misconceptions among 10- to 12-year old primary school children in Scratch using three different instruction methods. A mixed design with both between- and within-subject factors was used to analyse the data. The within-subject factor is time, the participants were tested twice on their holding of programming misconceptions. The between-subject factors were the three instruction methods; worked examples, given subgoal labelling and self-explaining subgoal labelling. In the given subgoal labelling condition the subgoal labels were named and in the self-explaining subgoal labelling condition the participants must think about the names themselves. This is an experimental study in which data was collected using a multiple-choice questionnaire.

3.1 Participants

Participants were pre-selected if they were between 10- and 12-year old and were in 7th or 8th grade of a primary school in the Netherlands. Participants were included if they had a command of the Dutch language; this was measured by asking about the reading level in the questionnaire. The reading level must be at least E6, which is the average for 9- to 10-year old children in the Netherlands. It was assumed that children from 10- to 12-year old work well independently on a school assignment. Participants did not get any financial compensation, they only received three free classes in Scratch. Because the participants were under 16 years old, permission was obtained from the parents/guardians by using a consent form before the study. Parents/guardians or participants themselves could stop participating in the study at any time.

Due to the COVID-19 outbreak and lockdown, it was hard to find participants. Also, the reopening of the schools during the research period resulted in several dropouts since children suddenly had no time for the programming classes. In the end, the sample consisted of 23 participants. Four participants had to be excluded – since one of them did not perform the tasks as requested and three children stopped after the first class because of a lack of time – leaving a final sample size of 19 participants (11 female; M_{age} = 10.74). The worked example condition had five participants, the given subgoal labelling condition had eight participants and the self-explaining subgoal labelling condition had six participants

Because of the small sample size, it is more useful to see this research as a pilot study which influenced the interpretation of the results. Because pilot studies are generally underpowered it is hard to achieve statistical difference at the 5% level (Lee, Whitehead, Jacques, et al, 2014). Therefore, the P-values described in the results should be interpreted with the knowledge that this study is not adequately powered (Lee, Whitehead, Jacques, et al, 2014). However, analysing effect sizes and descriptives can show the size and direction of the effect from instruction methods (Lee, Whitehead, Jacques, et al, 2014) which can be useful for further research.

3.2 Design

The study used an experimental design with a between-subject factor (instruction methods) and a within-subject factor (time). The study used two tests to measure the misconceptions. The first test took place after the first class and the second test took place before the third class. The first test was performed after the first class because misconceptions cannot be found without participants having knowledge about the concepts. The second test was used to test the retention of the programming concepts. This study design allowed the study to explore the one-week retention of programming misconceptions after three different instruction methodologies. The third class was not relevant for the research but helped to keep the children motivated to complete the questionnaire. During the online classes, the participants received the instructions based on the condition they were randomly assigned to. The participants stayed in the same condition during the whole research. See Figure 2 for a visual representation of the study design.

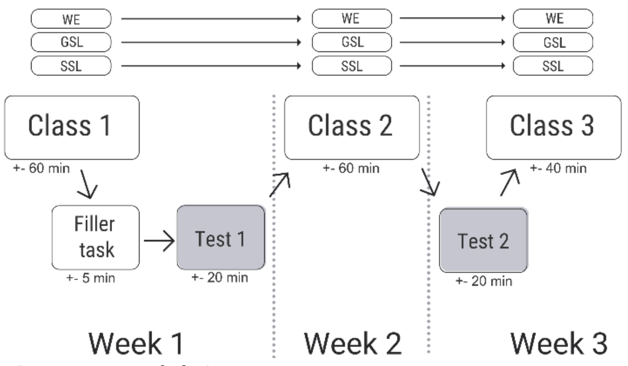


Figure 2: Research design
Note: WE = Worked example, GSL = Given subgoal labelling and SSL = Self-explaining subgoal labelling

3.3 Procedure

Participants were selected by convenience sampling. Social media posts were shared with the message that children could sign up for three free programming classes in Scratch. The message was shared via the newsletter from Programming Education Research Lab (PERL) and the newsletter from Wetenschapsknooppunt, an organization for education about science and technology. One primary school in Amsterdam shared this message with their students. The message contained a link where people could leave contact details. After filling in the contact details, they were contacted to schedule the three classes. The participants were randomly assigned to one of the three conditions. To ensure equal sample sizes, block randomization in blocks of six was chosen.

Participants followed three online programming classes in Scratch. A day before the class they received a PDF file with the assignment. The instruction method of the assignment was based on their condition group. Ten minutes before the class started the participants received an invitation link to a video-call. The first class consisted of a fifteen-minute explanation about Scratch and took around 70 minutes in total. The first four steps were done together, to make sure all participants knew what to do. After the first class, the participants had to make a filler task and had to fill in the questionnaire to test the misconceptions. They had approximately five minutes for the filler task and fifteen minutes for completing the questionnaire. The second class took around 60 minutes, where in the beginning of the class, all learned blocks were shortly explained. After the explanation, the participants could work on the assignment independently. Before the third class started, the participants filled in the second questionnaire. Participants had access to the internet while completing the questionnaire but were instructed before not to use the internet. During all classes there was the ability to ask questions if something was unclear.

3.4 Materials

To measure misconceptions after different instruction methods, several materials were used. First, the misconceptions to be tested were chosen. The list of Sorva (2012) has been taken as a starting point to explore existing misconceptions. Misconceptions were included if they occurred in the literature once, were applicable for beginners and applied to Scratch. For example, misconceptions about return values are not selected since this concept does not occur in Scratch. To make the misconceptions applicable for beginners there is chosen to select the misconceptions for the topics: variables, loops, input, and if/else statements. In the end, 15 misconceptions were selected. Seven questions were about variables, five about if/else statements, two about loops and one about the input. The misconceptions used in this study are shown in Table 1.

Concept	**Misconception**
Variable	a variable can hold multiple values at a time
Variable	Primitive assignment works in opposite direction
Variable	A variable is a pairing of a name to a changeable value. It is not stored inside the computer
Variable	Primitive assignment stores equations or unresolved expressions
Variable	Assignment moves a value from a variable to another
Variable	the natural-language semantics of variable names affects which value gets assigned to which variable
Variable	Difficulties in understanding the sequentially of statements
If/Else	Code after if statement is not executed if the then clause is
If/Else	If statement gets executed as soon as its condition becomes true
If/Else	A false condition ends program if no else branch
If/Else	Both then and else branches are executed
If/Else	control goes back to start when condition is false
Loops	Adjacent code executes within loop
Loops	while loops terminate as soon as condition changes to false
Other	Difficulties understanding the effect of input calls on execution

Table 1: Tested misconceptions

3.4.1 Multiple-choice questionnaire

A multiple-choice questionnaire with exercises in Scratch was used to measure the misconceptions. This questionnaire was based on previous research (Swidan, Hermans, & Smit, 2018). A multiple-choice questionnaire is a widely used method to measure misconceptions (Ma, 2007) which makes it possible to quantitatively analyse the data. Although, the disadvantage of this is that errors can depend on motivation, reading errors or guessing (Tew, 2010). Therefore, questions with open answers are often used (Ma, 2007). With open questions more information can be retrieved about the emergence of misconceptions and new misconceptions can be explored. However, for this study the open questions are only used to test whether children have guessed their answer. Open questions have the disadvantage that they cannot be processed quantitatively. Besides, it influences the motivation of the children which can also influence the answers. In this study a multiple-choice questionnaire was chosen to make it possible to measure the misconceptions in a test that did not take too long and did therefore not influence the motivation of the participants. An open question has been added to briefly explain their answer. This does not force children to choose an answer which reduced the chance that children will guess. Every question had four possible answer options; a correct answer, a wrong answer, a misconception answer, and an option to fill in an answer.

The questionnaire consisted of 15 items where each question represented one misconception. The response for each misconception question has been analysed and the answers were labelled as 'misconception', 'correct' or 'wrong'. If someone filled in another answer, this answer was analysed and then categorised in one of the three categories. All materials during this study were in Dutch. The questionnaire for the first and second test differed slightly to prevent a learning effect from the test. The structure of the questions was the same, only the used data was different. Figure 3 shows a question that is used in the questionnaire from Swidan, Hermans, and Smit (2018)

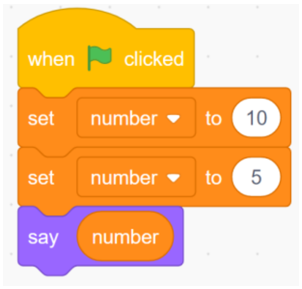


Figure 3: Example of the question that tests the misconception: 'a variable can hold multiple values at a time'

3.4.2 Programming classes

Three programming lessons were required for this study. Programming concepts, such as loops, variables, and if/else statements are often used together in one programming class, therefore participants only received one of the three conditions for all programming concepts. The classes from Felienne Hermans, obtained from https://scratchles.nl/ , were used with some additions to make sure all tested concepts were discussed in every class. Each lesson also had to be adapted to each of the three conditions.

For the worked example condition, the programming assignment was divided into steps. In the given subgoal labelling condition, the programming assignment was divided into steps, but the steps were also grouped and divided in subgoals with names. In the self-explaining subgoal labelling condition, the assignment was divided the same as the given subgoal labelling condition, only the names were left out. The participants were told to fill in the names for the subgoals themselves. The content was the same for all methods and was available for the children as a PDF document they could use during the class.

The concept of input occurred in the first class, but not in the second class. Because this could have influenced the number of misconceptions for this question, the topics were also analysed separately in the result section.

3.4.3 Filler task

A filler task between the class and the test was used to prevent the children from making the test based on memorizing steps. As an irrelevant filler task can prevent rehearsal of the criterium task (Houston 2014), a word search was chosen. Making a word search needs a high degree of concentration, which reduces the rehearsal process (Dillon & Reid, 1969). Also, a word search is different from programming which was important to avoid interference effects (Houston, 2014).

3.5 Statistical analysis

The data has been analysed in IBM SPSS statistics 26 for Windows. The data has been analysed by conducting a mixed ANOVA with time as a within-subject factor and instruction method as a between-subject factor. This test gave insight into the difference in the between-subject factor (instruction method), the within-subject factor (time) and the interaction of the within- and between-subject factors.

A sample size calculation with G * Power for a repeated measure ANOVA (two measure moments) with a between-subject factor (three instruction methods) was conducted and showed a desired sample of 120 participants for a power of 0.8 and an effect size of 0.25. The effect size is difficult to determine because little research has been done into programming misconceptions among children. For this reason, the average effect size of 0.25 was chosen. Each group of instruction method required 40 participants. This is a minimum desired number, based on an optimal situation.

The study situation was not optimal during the research. Schools were closed and lessons were cancelled due to measures related to the COVID-19 virus. Due to the pandemic and initiated lockdown children were not allowed to go to school. The research design changed from physical classes in schools to online classes at home. This made it impossible to achieve the desired sample size. It is difficult to reach children from this target group and, additionally, the COVID-19 virus causes a lot of change for the children. Also, because children could not be tested in a classroom setting and were tested alone or in pairs, data collection took more time.

4. RESULTS

4.1 Misconceptions

Figure 4 shows the distribution of answers over the three answer categories in both tests.

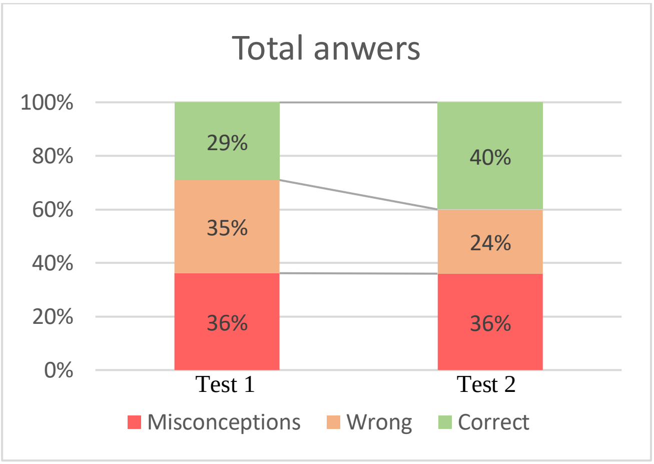


Figure 4: Distribution of answers

Taking all conditions together it can be seen that there is no difference in the number of misconceptions between the two tests. Wrong answers seem to decrease because students give more correct answers on the second test.

Nine participants (47%) had five or fewer misconceptions in the first test. In the second test nine participants (47%) had four or fewer misconceptions.

4.1.2 Relation misconceptions and instruction method

	Condition	N	Mean	Std. Deviation	Min	Max
Test 1	*WE*	5	5.80	1.78	3	7
	GSL	8	5.87	2.41	2	9
	SSL	6	4.50	2.43	0	7
Test 2	*WE*	5	6.60	2.41	4	9
	GSL	8	5.13	2.30	2	8
	SSL	6	5.00	1.79	3	8

Table 2: Misconceptions

Table 2 shows the means of the number of misconceptions in each condition and Figure 5 shows a visual representation of the means of misconception in each condition in both tests.

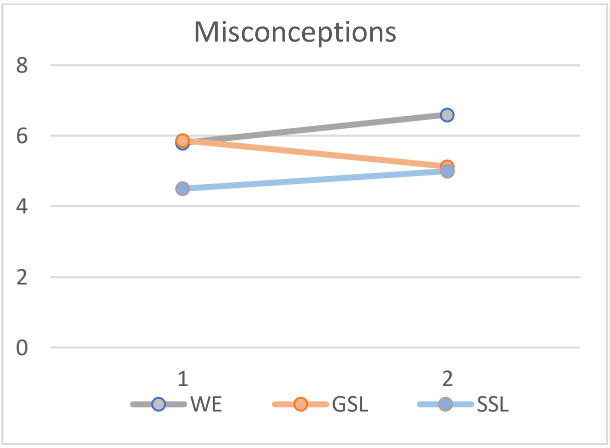


Figure 5: Means for the number of misconceptions in the first and second test

There is a small difference, of one and two misconceptions, between the conditions. The given subgoal labelling appears to be the highest in the first test (M=5.87, SD=2.41) closely followed by the worked example (M=5.80, SD=1.78). The self-explaining subgoal labelling condition had the lowest mean in both tests (M1=4.50, SD=2.43; M2=5.00, SD=1.79). The mean for the number of misconceptions in the given subgoal labelling condition is the only one that decreased over time. By comparing the means, the self-explaining subgoal labelling condition seems to be the most effective instruction method for avoiding, but not for decreasing misconceptions.

Performing a mixed ANOVA with time as a within-subject factor and instruction method as a between-subject factor, shows no significant result in the main effect of misconceptions over time F (1, 16) = 0.07, η_p^2 = 0.00, P=0.797). So, on average the number of misconceptions did not vary over time. Also, no significant effect in the main effect of instruction method F (2, 16) = 1.06, η_p^2 = 0.12, P=0.370) is found which means there is on average no difference in the number of misconceptions per instruction method. Although no significant difference is found, the effect size may suggest there is a small effect of instruction method on the number of misconceptions (η_p^2 = 0.12). The self-explaining subgoal labelling seems to have the most positive effect on the number of misconceptions and the worked example condition had the highest number of misconceptions. The interaction between instruction method and time F (2, 16) = 0.51, η_p^2 =0.06, P=0.611) also did not show a significant result. So, on average no difference in effect of misconceptions over time for different instruction methods is found. These results show no difference in the number of misconceptions that retain after one week of using different instruction methods among primary school children is found.

Performing a one-sample T-test shows 95% CI [4.34, 6.50] for the number of misconceptions in the first test and 95% CI [4.43, 6.52] for the number of misconceptions in the second test. Table 3 shows the 95% confidence intervals for each instruction method. The interval was not very precise because of the small sample size. However, looking separately at the conditions, the confidence interval in the second test became narrower which means there is less difference between the number of misconceptions among the different participants. It also shows that all means move to lower values over time.

	Condition	Lower bound	Upper bound
Test 1	WE	3.69	7.92
	GSL	3.08	6.42
	SSL	3.40	7.26
Test 2	WE	2.59	5.41
	GSL	2.26	4.49
	SSL	2.05	4.62

Table 3: 95% Confidence interval

4.2.2 Relation between misconceptions and topic

Figure 6 shows the labelled answers for the topics in the first and second test. Taking all the conditions together it can be seen that the instruction methods overall had the most negative effect on decreasing the number of misconceptions about loops and the most positive effect on decreasing the number of misconceptions about variables. The input question had an increase of 10% in the number of misconceptions, although it cannot be proven that this is because the concept did not occur in the second lesson. The misconceptions about loops are also increased, and this concept has been addressed in both lessons. Wrong answers decrease in all topics, sometimes they decreased more than the increase of the correct answers. This means that wrong answers do not always turn into a correct answer but can also turn into a misconception over time.

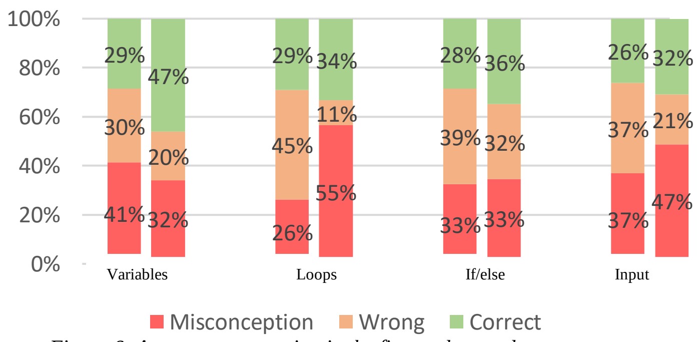


Figure 6: Answers per question in the first and second test

4.2.3 Misconceptions per topic and condition

There might be a relation between the number of misconceptions in a topic and instruction method and therefore the data of misconceptions in a topic is analysed in each condition. It may be that some instruction methods work better or worse for topics. Table 4 shows the means and maximum for the number of misconceptions in each condition.

	Condition	Variables	Max	If/else	Max	Loops	Max	Input	Max
Test 1	WE	2.60	*5*	2.00	*3*	0.80	*2*	0.40	1
	GSL	3.38	*5*	1.63	*3*	0.50	*1*	0.38	1
	SSL	2.89	*4*	1.63	*2*	0.53	*1*	0.33	1
Test 2	WE	2.80	*4*	2.20	*3*	1.20	*2*	0.40	1
	GSL	2.38	*5*	1.38	*2*	0.88	*2*	0.50	1
	SSL	2.26	*3*	1.63	*2*	1.11	*2*	0.50	1

Table 4: Mean and maximum number of misconceptions in each condition
Note: Variables consisted of seven questions, if/else statements of five questions, loops of two questions and input of one question.

In the worked example condition, the misconceptions in all topics increased and was therefore not effective in avoiding or reducing misconceptions about programming concepts. However, the effect of the condition was a more consistent line than the subgoal labelling conditions, which both showed larger differences in increase or decrease between the different topics. If people go from an incorrect understanding of the concept to a correct understanding, or the other way around, they have changed their knowledge about a concept. These results suggest that participants in the subgoal labelling conditions change their understanding of the concepts more.

A remarkable result is that the misconceptions about loops are increased in all conditions, which suggest that all instruction methods are equally ineffective for gaining correct understanding about the concepts of loops. The misconceptions about the input, also increased in each condition but since this topic did not occur in the second class the same conclusion cannot be drawn.

A mixed ANOVA with time as a within-subject factor and instruction method as a between subject factor showed for the main effect of instruction method on the number of misconceptions within variables F (1, 16) = 0.72, η_p^2=0.08, P=0.501), if/else statements F (2, 16) = 3.10, η_p^2=0.05, P=0.073), loops F (2, 16) = 0.74, η_p^2 =0.08, P=0.495) and input F (2, 16) = 0.22, η_p^2 = 0.08, P=0.978) no significant differences. So, on average, there was no difference in the number of misconceptions in each topic per instruction method. The interaction between time and instruction method for variables F (2, 16) = 0.47, η_p^2= 0.06, P=0.634), if/else statements F (2, 16) = 0.58, η_p^2=0.28, P=0.640), loops F (2, 16) = 0.497, η_p^2 = 0.06, P=0.618) and input F (2, 16) = 0.06, η_p^2 = 0.06, P=0.946) also did not show any significant results. So, on average, there was no difference in effect of instruction method on misconceptions over time in each topic.

Due to the small sample size, it is not unexpected that there were no significant differences. Looking at the effect sizes, showed that most effect sizes were very small, between η_p^2=0.05 and η_p^2=0.08. However, the partial eta squared showed a bigger effect of instruction method on the misconceptions about if/else statements (η_p^2=0.28). This suggest that the instruction method had the highest effect on the number of misconceptions about if/else statements.

4.3 Wrong and correct answers

Because no effect was found between instruction method and the number of misconceptions, we take a closer look at correct and wrong answers. This can be relevant as the number of misconceptions can be affected by wrong (no knowledge about the concept) or correct answers (correct knowledge about the concept). The number of correct answers is increased with 11%, the number of wrong answers is decreased with 11% and the number of misconception answers stayed the same. In 67% of the questions the number of wrong answers was decreased over time. Probably most of these wrong answers have been turned into correct answers since there is a high increase of correct answers over time. Also, in 67% of the questions the number of correct answers is increased over time. Table 5 shows the means for the number of wrong and correct answers in each condition.

	Condition	**Mean Wrong**	**Std. Deviation**	**Mean Correct**	**Std. Deviation**
Test 1	*WE*	5.80	*2.78*	3.40	2.07
	GSL	4.75	*2.25*	4.38	3.50
	SSL	5.33	*1.63*	5.17	3.00
Test 2	*WE*	4.40	*1.87*	4.40	3.37
	GSL	3.38	*1.51*	6.50	2.51
	SSL	3.33	*1.03*	6.67	1.97

Table 5: Means for number of correct and wrong answers

The means for correct answers in every condition was increased over time. The self-explaining subgoal labelling condition had in the second test the lowest mean for wrong answers (M=3.33, SD=1.03). In addition, the mean for correct answers was the highest in this condition (M=6.67, SD=1.97). The worked example condition had in the second test the highest mean for the number of wrong answers (M=4.40, SD=1.87) and the lowest mean for the number of correct answers (M=4.40, SD=3.37).

4.3.1 Wrong answers

Performing a mixed ANOVA with time as a within-subject factor and instruction method as a between-subject factor showed a significant difference in the main effect of time on wrong answers F (1, 16) = 7.22, η_p^2 = 0.31, P=0.016). This suggests each method to be effective for decreasing the wrong answers. There is no significant difference in the main effect of instruction method on wrong answers F (2, 16) = 0.64, η_p^2 = 0.07, P=0.542) nor the interaction effect F (2, 16) = 0.10, η_p^2 = 0.01, P=0.910). So, no difference in number of wrong answers per instruction method is found and also the effect of instruction methods on wrong answers over time did not differ in each condition. The partial eta squares also suggest that there is an effect of instruction method on the number of wrong answers over time (η_p^2 = 0.31) but the main effect of instruction method on the number of misconceptions was very small ($\eta_p^2 = 0.07$). The mean for number of wrong answers decreased in each condition over time and had the highest decrease in the self-explaining subgoal labelling condition. This suggest the self-explaining subgoal labelling condition to be the most effective for decreasing wrong answers about programming concepts over time.

4.3.2 Correct answers

Besides wrong answers, the number of misconceptions can also be affected by the number of correct answers. A mixed ANOVA with time as a within-subject factor and instruction method as a between-subject factor showed no significant difference on the main effect of time on correct answers F (1, 16) = 3.66, $\eta_p^2 = 0.19$, P=0.074) nor on the main effect of instruction method F (2, 16) = 1.99, $\eta_p^2 = 0.13$, P=0.330). Also, in the interaction of time and instruction method, no significant difference is found F (2, 16) = 0.17, $\eta_p^2 = 0.02$, P=0.845). No relation is found between the instruction method and number of correct answers over time. The partial eta squares for correct answers suggest there is a small effect for correct answers over time ($\eta_p^2 = 0.19$). Which suggest all instruction methods to be effective for increasing correct answers. Also, a small effect size was found in the main effect of instruction method ($\eta_p^2 = 0.13$). The self-explaining subgoal labelling condition had the highest mean for correct answers in both tests, the worked example condition had the lowest mean for correct answers in both tests. These results suggest the subgoal labelling conditions to be more effective in gaining a correct understanding of programming concepts.

5. DISCUSSION

In this study, the effect of different types of instruction methods in classes where 10- to 12-years old children were learning to program in Scratch was assessed. The one-week retention of misconceptions was measured using a multiple-choice questionnaire with exercises in Scratch. The aim was to study if there was a difference in the number of misconceptions using different instruction methods. The study used three different methods of instruction that have not yet been explored in Scratch to test the retention of misconception: worked example, given subgoal labelling and self-explaining subgoal labelling.

No significant difference between the instruction method and retention of programming misconceptions in Scratch was found in this study. On average, the number of misconceptions did not vary over time or per instruction method. Also, no effect of misconceptions over time for different instruction methods is found. However, due to the COVID-19 virus, the schools were closed which resulted in a small sample size. Since little research is done on these instruction methods in programming education in Scratch with children this research was relevant as a pilot study. This affected the interpretation of the results, the P-value was not the most important. So, looking at the descriptives and effect sizes suggests there is a small effect of instruction method on the number of misconceptions ($\eta_p^2 = 0.12$). This suggests further research can be relevant.

5.1 Misconceptions and instruction method

Looking at the descriptives of all conditions together showed no difference in the number of misconceptions between the two tests. However, it showed a small difference, of one and two misconceptions, between the three conditions. Also, a small effect size is found between the number of misconceptions and instruction method ($\eta_p^2 = 0.12$) which suggest there is a difference between the number of misconceptions between the conditions. The self-explaining subgoal labelling condition had the least number of misconceptions in both tests. The mean for misconceptions in the given subgoal labelling condition in the first test was higher than the worked example condition. However, the mean was in the second test together with the self-explaining subgoal labelling between one and two misconceptions lower than the worked example condition. Only in the given subgoal labelling condition there was a decrease of misconceptions which suggest this instruction method to be the most effective in reducing misconceptions over time. The self-explaining subgoal labelling condition had in both tests the lowest mean for the number of misconceptions which suggest this instruction method to be effective for avoiding misconceptions.

Both subgoal labelling conditions had fewer misconceptions compared to the worked example condition. This is in line with the literature that suggested subgoal labelling to be beneficial while programming (Joentausta & Hellas 2018). Children in the worked example may perform less because it was hard to understand how to solve a problem without having subgoals which help to distinguish the key features from the worked example (Catrambone, 1998). The self-explaining subgoal condition had the least misconceptions in both tests, but the misconceptions increased over time. One possibility could be that children in the self-explaining subgoal labelling condition had to interpret the knowledge by themselves, while the given subgoal labels sends the learning in a direction of interpreting the knowledge. Existing knowledge is important when learning new knowledge (Stomp & Schoenmaker, 2002). When the existing knowledge is incorrect or the given information is incomplete the new information can be placed in the wrong context (Hollingsworth & Ybarra, 2017) and a misconception can occur. Because novices do not have much existing knowledge it may be that more information such as the subgoal is needed to place the new information in the right context. The high space for interpretation in the self-explaining subgoal labelling condition could have caused to misinterpret the knowledge which can cause misconceptions. This could also have resulted in different interpretations during different classes.

5.2 Misconceptions in each topic

After finding small differences in the number of misconceptions for all topics together, the misconceptions were also analysed per topic. This ensures that all topics can be analysed separately, also the question about the input concept which did not occur in the second class. It may be possible that each topic prefers to be explained with another instruction method. Although, no significant results were found, some small differences in the descriptives and effect sizes could be noticed. The misconceptions for each topic increased after using worked examples, and also the means for number of misconceptions in each topic were the highest in this condition. So, worked examples did not seem to be effective in reducing programming misconceptions over time. There was a medium effect ($\eta_p^2 = 0.28$) found between instruction method and misconceptions in if/else statements. This may suggest that the understanding of if/else statements was affected by the used instruction method. Descriptives showed that the misconceptions about loops were increased in all conditions over time, which may suggest the concept of loops needs less independently explaining.

There can be a preferred difference for instruction method between the topics because some topics can be more difficult to learn than others. Also, it may be that some topics, such as loops, have a high change of misinterpreting the concept in an independently learning method while for some topics, it is clearer how to interpret the concept. This may explain the different results between the topics.

Although the worked example condition did not seem effective for reducing misconceptions, participants were most consistent in the number of misconceptions for a specific topic. In the subgoal labelling conditions there was more change between the number of misconceptions in the topics, which suggested children in these conditions changed their understanding about the concepts more. This can be the result of children being more aware of the steps they are taking while working with subgoals. Changing knowledge is important to overcome a misconception because misconceptions occur by an incorrect understanding.

5.3 Wrong and correct answers

Wrong answers seem to decrease because students give more correct answers in the second test. The subgoal labelling conditions seems the most effective because there were fewer wrong answers compared to the worked example condition. The self-explaining subgoal labelling instruction method was the most effective for learning programming concepts because this condition had the least misconceptions and wrong answers and the most correct answers. This may be because children are more aware of the steps they were taking which resulted in a correct understanding of the knowledge. This is in line with literature suggesting that generating or producing additional knowledge beyond the given instructions is more effective for learning compared to only following and repeating the given instructions (Chi, 2009).

Because no difference was found in misconceptions all together over time and this difference was found for wrong answers over time (P=0.016), it suggests that misconceptions are more tenacious compared to wrong answers. When having a misconception people think they have the correct understanding of the concept, so they do not know from themselves that they have a misconception which may cause misconceptions to be more tenacious. Previous research already suggested that misconceptions can remain for a long time (Simon, 2011) and maybe a one-week time period was not enough time to change misconceptions into correct answers. It is possible that the children need more practice with the concepts to realise their knowledge about a concept is incorrect.

5.4 Challenges

Besides the lockdown during the research, which made finding participants hard, the study faced some other challenges. First, since the classes could not be given in schools, due to the COVID-19 virus, the classes were given online. Children experienced much change during the time period of this research. In combination with following the classes at home could have influenced their learning ability. The children also followed the classes during different times during the day, something on a free day and sometimes after a school day which can have affected their performance. In the online classes is was harder to explain things. Explaining things took longer because some children first needed explanation of how to share their screen. Also, sometimes technological problems occurred and children had to reopen the video-call. During the online classes there was no break which may have resulted in tired and less motivated children at the end of the class. Because motivation results in better learning (Castejón, Gilar & Pérez, 2006; Schunk & Zimmerman, 2012) this can have influenced the questionnaire after the first class. However, this research is done with children that participated with intrinsic motivation. It can therefore be assumed that motivation did not negatively influenced the results. Also, all conditions followed the same classes which makes it unlikely that huge differences between the conditions occurred.

Also, the use of a multiple-choice questionnaire had its limitation. Although there was an open question to explain their answer, not many children responded to this question. It may be due to the many questions, which resulted in children needing a long time to complete. Completing did not take longer than expected, but the combination of no break and the online lesson may have made the children less motivated to answer all open questions. Also, in the first class many children did not understand certain concepts and filled in 'I don't know'. This resulted in many wrong answers which affected the number of misconceptions. The answers in the misconception and wrong category were both wrong answers, but the difference is that people with a wrong answer mostly did not understand the concept and know that they have the incorrect understanding and with a misconception they think they understand the concept correctly. This makes it harder to change a misconception answer into a correct answer. Fewer misconceptions do not always mean that children understand the programming concepts better, it may be that children have few misconceptions, but many incorrect answers. This means that the number of misconceptions cannot be taken as a measurement to see how well children perform on learning programming concepts.

One programming concept occurred in the first lesson, but not in the second lesson: the concept of the input in Scratch. Analysing the data showed an increase in the number of misconceptions about this concept, which may be influenced by the fact that in the second class the children did not work with it and forgot about it. However, the misconceptions about loops were also increased and this concept did occur in the second class. Also, the question about the input concept decreased in wrong answers and increased in correct answers, which is not different from the other topics. Based on these results, no different difference is found between practice and the retention of the programming concepts.

5.5 Further research

To make recommendations to the practice, a larger sample size is needed. Also, a more consistent study environment where children can be tested in schools is desirable. More children can be tested simultaneously, children are less distracted from their own home environment and it is easier to supervise on how the children are doing with the assignment.

Because this study suggest misconceptions are more tenacious, testing misconceptions over a longer term, such as one year, would be relevant. After a long time period, children would have a fully (in)correct understanding of the concepts which may result in more misconceptions. In the current research, it could be possible that children had not enough knowledge about some concepts to have a misconception. For this study, long-term research was not feasible. Programming is not integrated into the primary school's curriculum in such a way that programming is taught throughout the year. That is why long-term research in primary schools is difficult.

The effect sizes and descriptives suggest there may be small effects of instruction method on misconceptions and correct answers and therefore further research can be relevant. Since children in the subgoal labelling conditions performed better, it can be interesting to focus on these methods instead of the worked example in further research. Because it seemed like children in these two conditions changed their mind more about concepts, more support such as feedback or hints may be relevant. In the self-explaining subgoal labelling condition children may benefit from feedback or a list of possible subgoal names from which they can choose. Feedback allows children to know whether they interpret a concept correctly. This may prevent the number of misconceptions from increasing in the self-explaining subgoal labelling condition. Feedback can also help in how to interpret knowledge if they do not understand a concept and may therefore also decrease the number of wrong answers more. Also, it is interesting to investigate if the given labels from children can be used to detect misconceptions faster or can be used to give insight into the interpretation of programming concepts. This could both help in avoiding or reducing misconceptions.

Another interesting subject for further research with the self-explaining subgoal labelling can be to discuss the labels with each other. Previous research suggested that students change their knowledge about concepts faster if they discuss them with fellow students compared to a teacher telling them the right knowledge about the concept (Stomp & Schoenmaker, 2002). For children it may be interesting to let them conduct the subgoal labels and let them discuss their labels with each other and compare this with them discussing with a teacher.

References

Atkinson, R. K., Derry, S. J., Renkl, A., & Wortham, D. (2000). Learning from examples: Instructional principles from the worked examples research. Review of educational research, 70(2), 181-214.

Armoni, Meerbaum-Salant, and Ben-Ari. 2015. From Scratch to “Real” Programming. ACM Trans. Comput. Educ. 14, 4, Article 25 (February 2015), 15 pages.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (Eds.) (2000). How people learn: Brain, mind, experience, and school: Expanded edition. Retrieved from http://www.nap. edu/catalog.php?record_id=9853

Castejon, Juan-Luis & Gilar, Raquel & Perez-Sanchez, Antonio. (2006). Complex learning: The role of knowledge, intelligence, motivation and learning strategies. Psicothema. 18. 679-85.

Catrambone, R. (1998). The subgoal learning model: Creating better examples so that students can solve novel problems. Journal of experimental psychology: General, 127(4), 355.

Chi. M. T. H. (2009). Active-constructive-interactive: A conceptual framework for differentiating learning activities. Topics in Cognitive Science, 1(1), 73-105.

Chi, M. T., & Wylie, R. (2014). The ICAP framework: Linking cognitive engagement to active learning outcomes. Educational psychologist, 49(4), 219-243.

Cooper, G., & Sweller, J. (1987). The effects of schema acquisition and rale automation on mathematical problem-solving transfer. Journal of Educational Psychology, 79, 347-362

Dillon, R. F., & Reid, L. S. (1969). Short-term memory as a function of information processing during the retention interval. Journal of Experimental Psychology, 81(2), 261–269.

Du Boulay, B. 1986. Some Difficulties of Learning to Program. Journal of Educational Computing Research 2, 1 (1986), 57–73.

Firdaus, F. M. (2017). Improving Primary Students' Mathematical Literacy through Problem Based Learning and Direct Instruction. Educational Research and Reviews, 12(4), 212-219.

Gomes, Anabela & Mendes, Antonio. (2007). Learning to program - difficulties and solutions. 283-287.

Hermans, F., & Aivaloglou, E. 2017. Teaching Software Engineering Principles to K-12 Students: A MOOC on Scratch. In Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track. 13–22.

Hermans, F., & Smit, M. (2018). Explicit Direct Instruction in Programming Education. In Proceedings of the 29th Annual Conference of the Psychology of Programming Interest Group (PPIG 2018) (pp. 86-93).

Hollingsworth, J. R., & Ybarra, S. E. (2017). Explicit direct instruction (EDI): The power of the well-crafted, well-taught lesson. Corwin Press.

Houston, J. P. (2014). Fundamentals of learning and memory. AcademicPress.

Ichinco, M., Harms, K. J., & Kelleher, C. (2017). Towards understanding successful novice example user in blocks-based programming. Journal of Visual Languages and Sentient Systems, 3, 101-118.

Joentausta, J., & Hellas, A. (2018, February). Subgoal labeled worked examples in K-3 education. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (pp. 616-621).

Kalyuga, S., Chandler, P., Tuovinen, J., & Sweller, J. (2001). When problem solving is superior to studying worked examples. Journal of educational psychology, 93(3), 579.

Klahr, D., & Nigam, M. (2004). The Equivalence of Learning Paths in Early Science Instruction: Effects of Direct Instruction and Discovery Learning. Psychological Science, 15(10), 661–667

Lee, E. C., Whitehead, A. L., Jacques, R. M., & Julious, S. A. (2014). The statistical interpretation of pilot trials: should significance thresholds be reconsidered?. BMC medical research methodology, 14(1), 41.

Ma, L. (2007). Investigating and Improving Novice Programmers’ Mental Models of Programming. Concepts. University of Strathclyde

Margulieux, L. & Catrambone, R. (2016). Using Subgoal Learning and Self-Explanation to Improve Programming Education.

McLaren B.M., van Gog T., Ganoe C., Yaron D., Karabinos M. (2014) Exploring the Assistance Dilemma: Comparing Instructional Support in Examples and Problems. In: Trausan-Matu S., Boyer K.E., Crosby M., Panourgia K. (eds) Intelligent Tutoring Systems. ITS 2014. Lecture Notes in Computer Science, vol 8474. Springer, Cham

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. Computer Science Education, 23(3), 239-264.

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.

Ouahbi, İ., Kaddari, F., Darhmaoui, H., Elachqar, A., & Lahmine, S. (2014). Serious Games for teaching combined basic programming and English communication for non-science major students. International Journal on Advances in Education Research, 1(1), 77-89.

Paas, F. G. (1992). Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive-load approach. Journal of educational psychology, 84(4), 429.

Price, T. W., & Barnes, T. (2015, August). Comparing textual and block interfaces in a novice programming environment. In Proceedings of the eleventh annual international conference on international computing education research (pp. 91-99).

Rahman, S. S. A., & du Boulay, B. (2010). Learning programming via worked-examples. PPIG-WIP, Dundee 2010, 1-6.

Schunk, D. H., & Zimmerman, B. J. (Eds.). (2012). Motivation and self-regulated learning: Theory, research, and applications. Routledge.

Simon. 2011. Assignment and sequence. Proceedings of the 11th International Conference on Computing Education Research (2011).

Sorva, J. 2008. The same but different students’ understandings of primitive and object variables. Proceedings of the 8th International Conference on Computing Education Research (2008).

Sorva, J. 2012. Visual program simulation in introductory programming education. PhD Thesis, Aalto University. (2012).

Stomp, Lex & Schoenmaker, Karel & valstar, johan. (2002). Van vakdidactiek naar een gedeelde opleidingsdidactiek. Velon tijdschrift voor lerarenopleiders. 2002. 14.

Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. Cognition and instruction, 2(1), 59-89.

Swidan, A., Hermans, F., & Smit, M. (2018, August). Programming misconceptions for school students. In Proceedings of the 2018 ACM Conference on International Computing Education Research (pp. 151-159).

Tew, A. E. (2010). Assessing Fundamental Introductory Computing Concept Knowdledge in a Language Independent Manner. Georgia Institute of Technology, School of Interactive Computing

Vihavainen, A., Airaksinen, J., & Watson, C. (2014). A Systematic Review of Approaches for Teaching Introductory Programming and Their Influence on Success. In ICER '14 Proceedings of the tenth annual conference on International computing education research (pp. 19-26). New York: ACM.

Zhi, R., Price, T. W., Marwan, S., Milliken, A., Barnes, T., & Chi, M. (2019, February). Exploring the impact of worked examples in a novice programming environment. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (pp. 98-104).