TU Delft

Delft University of Technology

# The influence of SRA programming on algorithmic thinking and self-efficacy using Lego robotics in two types of instruction

Fanchamps, Nardie L.J.A.; Slangen, Lou; Hennissen, Paul; Specht, Marcus

**DOI**
10.1007/s10798-019-09559-9

**Publication date**
2021

**Document Version**
Final published version

**Published in**
International Journal of Technology and Design Education

**Citation (APA)**
Fanchamps, N. L. J. A., Slangen, L., Hennissen, P., & Specht, M. (2021). The influence of SRA programming on algorithmic thinking and self-efficacy using Lego robotics in two types of instruction. *International Journal of Technology and Design Education*, *31*(2), 203-222. https://doi.org/10.1007/s10798-019-09559-9

**Important note**
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

**Copyright**
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

**Takedown policy**
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

*This work is downloaded from Delft University of Technology.*
*For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.*

International Journal of Technology and Design Education (2021) 31:203–222
https://doi.org/10.1007/s10798-019-09559-9

# The influence of SRA programming on algorithmic thinking and self-efficacy using Lego robotics in two types of instruction

**Nardie L. J. A. Fanchamps[1,2] · Lou Slangen[1] · Paul Hennissen[3] · Marcus Specht[2,4]**

Accepted: 13 December 2019 / Published online: 24 December 2019
© The Author(s) 2019

**Abstract**
This study investigates the development of algorithmic thinking as a part of computational thinking skills and self-efficacy of primary school pupils using programmable robots in different instruction variants. Computational thinking is defined in the context of twenty-first century skills and describes processes involved in (re)formulating a problem in a way that a computer can process it. Programming robots offers specific affordances as it can be used to develop programs following a Sense-Reason-Act (SRA) cycle. The literature provides evidence that programming robots has the potential to enhance algorithmic thinking as a component of computational thinking. Specifically there are indications that pupils who use SRA-programming learn algorithmic skills better and achieve a higher level of self-efficacy in an open, scaffold learning environment than through direct instruction. In order to determine the influence of the instruction variant used, an experimental research design was made in which pupils solved algorithm-based mathematical problems (grid diagrams) in a preliminary measurement and their self-efficacy determined via a questionnaire. As an intervention, pupils learn to solve programming issues in pairs using "Lego NXT" robots and "Mindstorms" software in two instruction variants. The post-measurement consists of a Lego challenge, solving mathematical problems (grid diagrams), and a repeated self-efficacy questionnaire. This research shows an increase of our measures on algorithmic thinking dependent on the amount of SRA usage (though not significant). Programming using the SRA-cycle can be considered as the cause of the measured effect. The instruction variant used during the robotic intervention seems to play only a marginal role.

**Keywords** Programming · Sense-reason-act · Robotics · Computational thinking · Mathematics · Self-efficacy

✉ Nardie L. J. A. Fanchamps
nardie.fanchamps@fontys.nl

[1] Fontys University of Applied Science, Mgr. Claessenstraat 4, 6131 AJ Sittard, The Netherlands

[2] Welten Institute (TELI), Open University, Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands

[3] Zuyd University of Applied Science, Nieuw Eyckholt 300, 6419 DJ Heerlen, The Netherlands

[4] Centre for Expertise and Learning, Delft University of Technology, Van Mourikbroekmanweg 6, 2628 XE Delft, The Netherlands

## Introduction

The impact of information technology, robotics and programming on our modern lives is tremendous. In order to be able to participate functionally in such a digital oriented society, it is necessary to become Information Communication Technology (ICT) literate (Kennisnet 2015; Maas 2015; Pelgrum 2001). For this reason, it is essential to allow young children to develop relevant ICT competencies (Aesaert et al. 2015). Research emphasizes the fact that primary education already has a responsibility to teach the required competences (Toh et al. 2016). Programming is one of these ICT-competencies and should be structurally embedded in primary education (Kazakoff et al. 2013).

Programming has been shown to have a positive effect on children's algorithmic thinking (Futschek and Moschitz 2011) and can play a powerful role by making concealed mathematical assumptions explicit and concrete (Kolovou et al. 2008; Silk et al. 2010; Wilensky 1995). Translating problem solving into a software program can help pupils to develop a deeper understanding of the working principles of algorithms, which in turn helps solving mathematical problems better. (Thijs et al. 2014; Kolovou et al. 2008). Understanding algorithmic thinking contributes to the development of computational thinking in general (Wing 2006).

Robotic contexts turn out to be powerful environments to learn computational thinking (Benitti 2012; Tay et al. 2014), since they make programming concrete and tangible (Horn et al. 2009; Wang et al. 2014). More specifically, programmable robots offer excellent opportunities to comprehend algorithmic solution strategies (Kolovou et al. 2008; Silk et al. 2010). The so-called Sense-Reason-Act (SRA) programming approach is based on the understanding of variable solution strategies that can play an important role in the development of computational thinking (Slangen et al. 2011b). By making pupils aware of the difference between the SRA-approach and a linear programming approach, pupils learn to understand that SRA can be a more efficient approach to finding a solution to complex programming problems.

Tangible programmable robots offer a Direct Manipulation Environment (DME) (Jonassen 2006) which makes the coherence between sensing (input), reasoning and acting (output) visible and understandable. Previous research shows that primary school pupils have difficulties in using SRA based reasoning (Slangen et al. 2011b). Even when pupils can use SRA programming, it is reasonable this usage requires a certain degree of self-efficacy (tenacity, self-responsibility and self-determination) to accomplish a difficult task in a self-directed way.

This research examines the influence of SRA-programming on algorithmic thinking as a component of computational thinking and the impact on self-efficacy among primary school pupils. The focus is on to what extent this development depends on the nature of teacher support and to what extent there is an increase on the self-allocated degree of self-efficacy. Therefore, it is investigated whether pupils with Lego Robotics NXT are able to construct a series of programming instructions in the correct order, with the result that a functioning algorithm is created.

## Theoretical framework

In this research we are particularly interested in the relationship between learning to configure robot control programs using a SRA-approach and the impact this has on algorithmic and computational thinking. We also want to know if the type of instruction used influences SRA-thinking and impacts the level of self-efficacy. From research we know that solving more advanced programming problems require more complex algorithmic

structures and only enables efficient problem solutions and programs if we go beyond linear programming (Slangen et al. 2011b; Wyeth et al. 2003). When programming following a SRA-approach users need to explicitly link the relationship between observations based on sensor use (*sense*), with a logic reasoning component which infers actions based on these observations (*reason*) and the process of acting based on the given inferences (*act*). Slangen (2016) examined the application of the SRA-concept in primary school practice using Lego Mindstorms robots and concluded that primary school pupils, although capable of programming robots, have considerable difficulties with understanding and applying the SRA-programming cycle. Mainly complex elements of programming like the 'if-then-else' and the 'nested loop' seemed to be difficult to understand. The ability to functionally apply the SRA-cycle in a robotics context requires pupils to apply logic reasoning in programming environments (Kurland et al. 1986; Pea and Kurland 1984). It also requires systematic thinking for a correct choice of sensors and actuators to program a robot that can actually anticipate the physical environment (Slangen et al. 2011b). SRA programming applied through robotic environments, provides meaningful opportunities to incorporate computational thinking skills and mathematics. SRA-programming is based on an algorithmic, mathematical approach and can be used to reformulate robotic problems. Moreover solving mathematical problems demands specific, algorithmic thinking skills to solve challenging computing and mathematical issues (Guzdial 2008) and stimulates analytical competences (Basawapatna et al. 2011).

Robotic-contexts provide relevant, interesting and challenging opportunities for the development of algorithmic thinking (King et al. 2012; Highfield et al. 2008). Silk and Schunn (2008) analysed a curriculum which provides a synergetic context for robotics, engineering and mathematics. They showed by means of content analyses, cases studies, and comparative studies, that unravelling robotics-based problems contributes to the skill of algorithmic thinking, which includes logical reasoning, argumentation, induction and deduction (Jacobs et al. 2010; Drijvers 2015; Schoenfeld and Sloane 2016). This also implies that algorithmic thinking can be taught by means of well-designed robotic-based environments which offers pupils the opportunity to activate and concretize such skills.

Computational thinking can enrich mathematics and vice versa, the integration of mathematical contexts can improve mathematical learning (Weintrop et al. 2016). Computational thinking is the processes involved in formulating a problem and expressing its solution(s) in such a way that a computer can effectively solve the problem (Barr et al. 2011). It is an iterative process based on three stages, i.e. problem formulation, solution expression and solution execution & evaluation (Yadav et al. 2016; Wong 2014; Wing 2006). Computational thinking refers to skills such as problem decomposition, pattern recognition/data representation, generalization/abstraction, and algorithmisation (Voogt and Roblin 2010). Through decomposing a problem, identifying the variables, using data representation and creating algorithms a generic solution results (Thijs et al. 2014). Robotic contexts provide the opportunity to apply components of computational thinking which facilitates the development of algorithmic thinking enabling the elucidation of underlying mathematical principles (Drijvers et al. 2009; Silk et al. 2010), i.e. understanding the core of an algorithm.

Teaching and learning how to program robotics depends on the instructional approach and is a mutual process between teachers and pupils (Bers et al. 2002). It is likely that variations in the pedagogical approach, for example the variation in the extent to which the teacher gives guidance on process and content, leads to different outcomes (Suárez et al. 2018). On the one hand we see guided instruction in which the initiative for setting out the next steps in the learning process lies with the teacher rather than with the

learner. On the other hand we also notice pedagogical approaches in which pupils learn by themselves through solving authentic problems with negligible teacher support. This form is also known as Project-based learning (PBL). Project-based learning is a teaching method in which pupils gain knowledge and skills by working for an extended period of time to investigate and respond to an authentic, engaging and complex question, problem or challenge (Blumenfeld et al. 1991; Kong 2008). It goes without saying that good teachers, depending on the pedagogical need of pupil-teacher interaction, can choose between a more guiding or coaching strategy. A teacher who understands the influence of the use of specific approaches in the teacher–pupil interaction is expected to make teaching more effective (Vosniadou et al. 2001). Therefore it is important that the teacher has a variety of coaching skills, intervention competences and dialog techniques at his disposal (Crasborn and Hennissen 2010).

Direct instruction can be defined as acts of the teacher with the aim to support the learning activities of pupils to structure those in a desired direction (Veenman 2001). The starting point for direct instruction is the assumption that there are moments in a teaching process when knowledge, insights and skills are most effective, meaningful, functional, and targeted when taught in a direct guided way to pupils (Kirschner et al. 2006). Direct instruction is particularly appropriate when a well-structured set of knowledge, insights and skills must be mastered by all pupils (Leenders et al. 2010).

Scaffolding can be defined as an approach that lets pupils learn more independently whereby the guidance provided by the teacher is temporary adapted to the level of pupil's understanding (Kapur and Bielaczyc 2012). It's a method used by the teacher to help the pupils to retrieve existing knowledge and to stimulate them to reach higher levels of comprehension and skill acquisition (Kawalkar and Vijapurkar 2011). When using scaffolding the teacher supports and guides pupils during the process when they are not capable by themselves or when the teacher notices that pupils are taking an entirely wrong direction and get stuck in the process (Hogan and Pressley 1997). Referring to the viewpoint of (Hmelo-Silver et al. 2007) the process to learn can be best organized in such a way that pupils are able to perform and solve problems independently. The teacher must ensure that he's not going to instruct however can use verbal scaffolding techniques whereby the thought process remains primarily by pupils.

It is important that the teacher is aware which impact the method of teaching used has on the learning outcomes of pupils (Slangen et al. 2008). Slangen (2016) addresses the question of whether pupils, when they learn to program, learn best when they carry out all the actions and steps themselves or whether the learning outcome is more extensive when the teacher constantly intervenes and explains everything to the pupils. Therefore, in this study two different instruction methods are used to determine which instruction variant shows the most optimal yield: direct instruction or scaffolding.

It is not yet clear why one person, in an introduction course, learns to program easily while another person under the same conditions experiences major difficulties (Hasan 2003). Despite instructions given by the teacher on how to program, learners often make completely autonomous decisions that deviate from the received instruction (Tay et al. 2014). It appears that the level of self-efficacy of the learner plays a decisive role in learning how to program (Igbaria and Iivari 1995). Bandura (1977) and Zimmerman (2000) describe self-efficacy as someone's belief in their own ability to complete a task successfully. Working autonomously and independently decision making contributes to the level of self-efficacy and self-effectiveness (Ramalingam and Wiedenbeck 1998). Previous research shows that primary schools pupils' self-efficacy for programming is influenced by previous programming experience and self-efficacy also increases as pupils progress through an

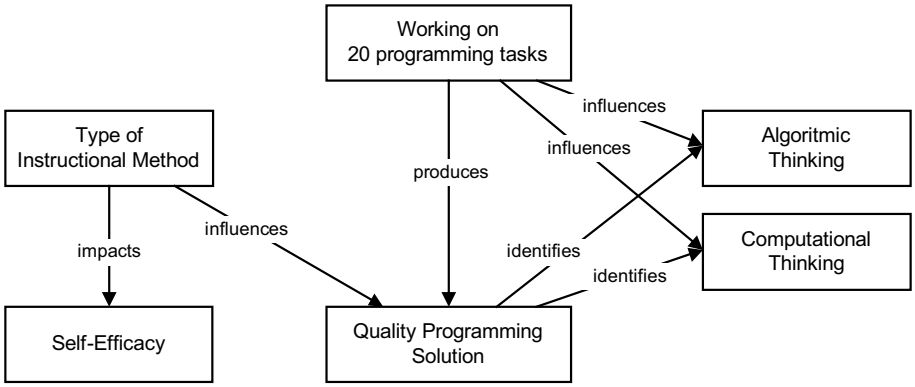


**Fig. 1** Schematic representation of the conceptual model

introductory programming course (Ramalingam et al. 2004) and showed positive effects on pupils perceived self-efficacy when learning to program Lego robotics (Liu et al. 2010).

Building on the theoretical exploration above, we presume a correlation between the SRA-approach, the instruction variant used with an impact on self-efficacy, the quality of programming e.g. more generic the influence on computational thinking or more specifically on algorithmic thinking. Therefore our conceptual model in Fig. 1 provides an overview of relationships and interconnections between the independent and the dependent variables.

## Research question, sub-questions and hypotheses

Based on the literature study we focus on investigating to what extent there is a relationship between problem-solving robotic programming, self-efficacy, computational thinking and algorithmic thinking.

In this research, the aim is to clarify the main research question: "What outcomes concerning algorithmic thinking and self-efficacy does SRA-programming in a Lego robotic context lead to when using a scaffolding approach or using a direct instructional approach?"

In addition to the main research question, the sub-questions are:

1. Can SRA-programming achieve a higher degree of algorithmic thinking?
2. Does SRA-programming lead to a higher degree of self-efficacy?
3. What is the influence of the instruction variant on the quality of the solution of the programming problem?

These sub-questions lead to the next three hypotheses:

1. Learning to apply SRA-programming in a Lego robotics context when using the instruction method scaffolding leads to a more successful level of solving mathematical grid diagrams.
2. Learning to apply SRA-programming in a Lego robotics context when using the instruction method scaffolding leads to a higher level of algorithmic skills.

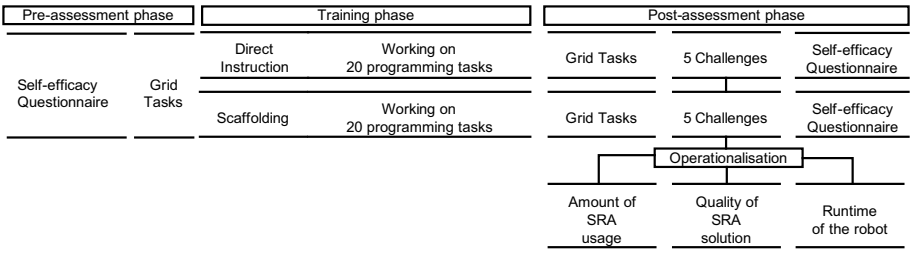


**Fig. 2** Research design

3. Using the instruction method scaffolding leads to a higher level of self-efficacy compared to using the direct instruction method.

## Method

This research should be seen as an explorative approach to gain more insight into what a robotics programming environment can contribute to the development of mathematical skill and self-efficacy. To this end, an exploratory study was conducted in which quantitative data was obtained to examine the research question and the associated hypotheses. To investigate the research questions and hypotheses we used a pretest–posttest design as displayed in Fig. 2. This includes (a) pre-measurement of mathematical skill and self-efficacy, (b) a robotics-intervention in two instruction variants, and (c) a post-test measuring the programming ability and quality, mathematical skill and self-efficacy.

### Participants

The research was conducted among pupils from grade 5 and grade 6[1] ($N=62$) of a primary school in the south of the Netherlands. An experimental group ($n=33$) and a control group ($n=29$) were formed. From these two subgroups, 31 equal strong pairs are compiled based on the average of both their individual mathematical score achieved on the Dutch Cito tracking scores for numeracy and mathematical skill (Cito 1987). In this way there are not very strong or weak subgroups so that this cannot obscure the results. Figure 3 displays the division of participating groups.

## Materials

Visually oriented programming environments such as Scratch and Lego Mindstorms are very suitable for use in primary education (Korkmaz 2018). No complicated text based code language needs to be learned and the user can drag and drop programmable blocks in the right order (Weintrop and Wilensky 2015). Visual programming environments are also ideally suited to control tangible objects such as robots (Sapounidis et al. 2015).

[1] In this publication we use the UK grade level system to indicate the research population. Grade 5 and 6 in the UK corresponds with the Dutch "group 7 and 8".

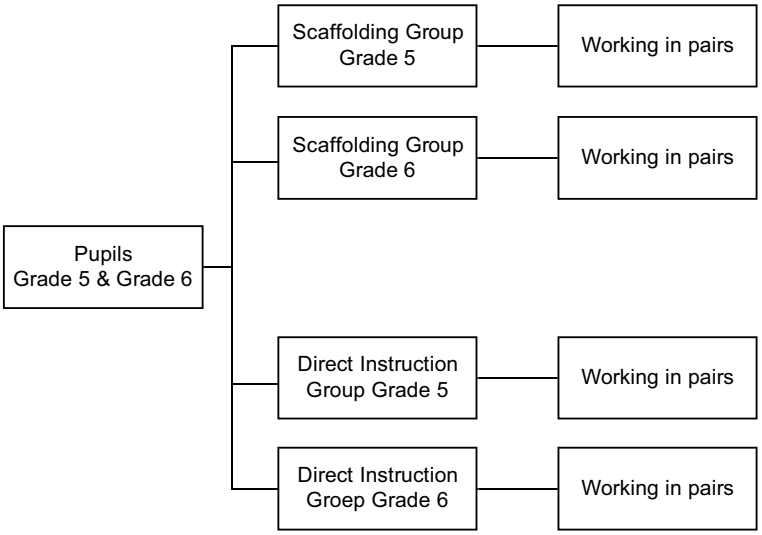


**Fig. 3** Grouping of scaffolding group and direct instruction group

Robots operate by a program which manages a programmable logical controller by means of using sensors and actuators (Sargent et al. 1996; Jonassen 2006; Lindh and Holgersson 2007). Benitti (2012) as well as Krumholz (1998) claim that the use of robots enhances learning. For instance, it provides opportunities to connect math, logical thinking and programming. Working with robots also offers hands-on experiences with immediate results giving the children both opportunities for creativity and a sense of achievement (Jeschke et al. 2008).

To determine the degree of self-efficacy in a pre- and post-measurement (prior and afterwards the robotic intervention), we used of the validated Dutch adaptation of the general self-efficacy scale (Teeuw et al. 1994) to which we have added two supplementary questions in order to be able to determine pupils' grade (5/6) and gender (male or female). Pupils completed this questionnaire individually. This self-efficacy questionnaire includes 10 items on a psychometric 4-point scale, ranging from 'completely inaccurate' to 'completely accurate', to indicate the extent to which the claim applies and as such is experienced. To determine the scale reliability, we calculated Cronbach's alpha. Noting that a value for Cronbach's alpha from 0.70 is considered an acceptable reliability factor (Santos 1999). As a characteristic, the developers of the this instrument indicate that Cronbach's alpha should range between $\alpha = 0.76$ and $\alpha = 0.90$ (Scholz et al. 2002; Schwarzer and Jerusalem 1995). We measured for Cronbach's alpha $\alpha = 0.77$ which indicates a high level of internal consistency for our scale with this specific sample. From this we can conclude that, despite the low value for *N,* the internal reliability of the adapted instrument used is sufficiently high and performs as reported by the original authors. The measurement results therefore can be used as such.

As a pre- and post-measurement of algorithmic skill, pupils individually solve six so-called grid diagrams (for examples: see Fig. 4). These diagrams, ascending in complexity, are based on the Pascal triangle (Barry 2006) where the number of shortest routes from A

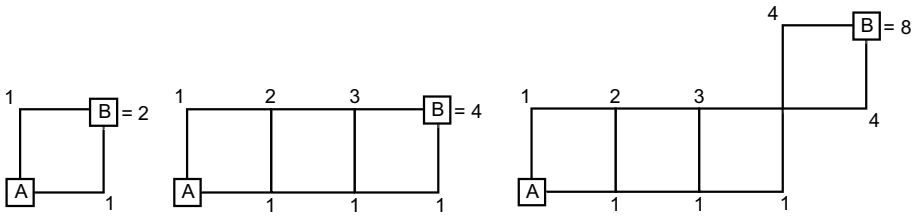


**Fig. 4** Grid diagrams; three examples

to B must be found. Solving these diagrams requires the application of a correct algorithmic approach in which grid diagram 1 is the easiest to solve and grid diagram 6 the most difficult.

Solving a grid diagram results in the following scoring options: grid diagram solved correctly (yes/no), algorithm applied (yes/no), algorithm constructed correctly (yes/no). The yield of the challenge consists of four categories: The first category (runtime needed) includes a fixed value, the second and third category (challenge solved, SRA applied) each has two values (yes/no), the fourth category (quality SRA), has three values ("SRA not applied"/"SRA multiple variations"/"SRA shortest possible").

As an intervention pupils in the control group and experimental group each receive a different instruction variant to learn SRA-programming Lego NXT Mindstorms robots. The Lego programming environment is based on the operating principles and routines of reasoning and decision making by programming visual blocks. These blocks represent specific activities such as sensing, reasoning, and acting via controllable variables, parameters, logical operators, et cetera. By manipulating the variables and sequencing the blocks in a specific order, pupils construct their program and conduct it. To demonstrate differences in programming ability and understanding of SRA we set up a pre-defined problem space in which pupils solved a Lego robotic programming challenge. This challenge, in which the influence of the applied instruction variant occurred, consists of a pre-made robot equipped with two different types of sensors mounted (push-button sensor and ultrasonic sensor) and a table as the playing field on which the robot has to execute the tasks. A successful programming solution and the fastest runtime out of three attempts is recorded. The runtime factor is taken as a predictive value and illustrates the degree of quality and efficiency of the constructed program.

The execution time needed for a computer program to complete its processing is an important indicator about the efficiency and quality of the constructed computer program (Korzilius 2000). In this study we follow the assumption that the less time a programmed robot needs to perform commands successfully, the more efficient the constructed computer program is (Ploeg 2015) which results in a higher level of quality (Brave et al. 2011). Because, computer programs are based on algorithms, time efficiency also determines if and how efficiently an algorithm is constructed.

## Procedure

All 31 pairs from both instruction variants were given identical introductory instructions explaining step by step how to use the Lego programming environment. Thereafter, each of the pairs conducted nine one-hour sessions to solve twenty programming tasks to

demonstrate the solution devised. In the direct-instruction variant we offered pupils during the nine-hour training sessions information and explanation on how to program a robot that makes use of sensors. In the scaffolding variant we coached pupils during the nine-hour trainings sessions by giving guidance adapted to pupils level of understanding on how to program a robot that makes use of sensors. In both instruction variants, pupils can decide for themselves whether to program without using sensors or programming with the fully functional and most effective use of sensors. In the final challenge-assignment pupils can show what they have learned from these programming tasks and what differences occur out of the two instruction variants.

## Results and data-analysis

The main research question, "What outcomes concerning algorithmic thinking and self-efficacy does SRA-programming in a Lego robotic context lead to when using a scaffolding approach or using a direct instructional approach?", is answered by analysing the means of the dichotomous variables. *T* test analysis is used to investigate sub-questions and to confirm or reject hypotheses. The self-efficacy questionnaire, the measured results to solve grid diagrams and the data derived from the Lego robotics challenge were entered into SPSS for quantitative data analysis. The effect of the independent variables on the dependent variables is investigated (see Fig. 3). Differences in values are determined by comparing the means. Cross-tabs are used to make a shift visible between pre- and post-measurement. A repeated measures analysis is used to determine the effect on self-efficacy. In all statistical analysis is assumed a significance level of 5% ($p = \leq .05$).

The nature of the data meets the conditions for the assumption of normality and asserts that the distribution of sample means (across independent samples) is normal. It has been tested whether the assumptions of homogeneity of variances have been violated ($p \leq 0.05$). Degrees of freedom are calculated and the bootstrapping procedure has been applied to re-estimate the standard error of the mean difference. The confidence interval was studied to assess the difference between means and to determine whether the value "0" is in the confidence interval. The value for the extent of the effect (Pearson's *r*) has been calculated (indicating that the effect size is low if the value of *r* varies around 0.1, medium if *r* varies around 0.3, and large if *r* varies more than 0.5). The substantial effect of a standard deviation difference between two groups (Cohen's *d*) was also determined (it should be noted that $d = 0.2$ can be considered a 'small' effect size, 0.5 stands for a 'medium' effect size and 0.8 for a 'large' effect size) (Field 2013).

### Mathematical problem solved correctly

A comparison of the total number of correctly resolved grid diagrams for the scaffolding group increases from 99 ($M = 0.51$) to 103 ($M = 0.55$) and for the direct instruction group increases from 88 ($M = 0.50$) to 96 ($M = 0.52$). Further examination of the data by means of cross-tabs analysis shows that more complex grid diagrams are solved in both groups. It is remarkable that in the pre-test no respondent correctly solves all six mathematical problems, while in the post-test 6 respondents do solve all six mathematical problems correctly. Table 1 shows the data for solving mathematical problems correctly.

**Table 1** Mathematical problem solved correctly

| | | Pre-test | | | | | | | | | Post-test | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Mathematical problem solved correctly | | | | | | | | | Mathematical problem solved correctly | | | | | | |
| Group | *n* | *M* | Total | 0 | 1 | 2 | 3 | 4 | 5 | 6 | *M* | Total | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Scaff. | 33 | 0.50 | 99 | 0 | 1 | 11 | 10 | 9 | 2 | 0 | 0.52 | 103 | 1 | 3 | 7 | 11 | 6 | 1 | 4 |
| Dir. | 29 | 0.51 | 88 | 0 | 4 | 4 | 10 | 9 | 2 | 0 | 0.55 | 96 | 0 | 3 | 1 | 17 | 3 | 2 | 2 |

Scaff., Scaffolding group; Dir., direct instruction group. M, Average number of resolved out of total; Total, Cumulative

**Table 2** Mathematical problem algorithm applied

| | | Pre-test | | | | | | | | | Post-test | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Mathematical problem algorithms applied | | | | | | | | | Mathematical problem algorithms applied | | | | | | |
| Group | *n* | *M* | Total | 0 | 1 | 2 | 3 | 4 | 5 | 6 | *M* | Total | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Scaff. | 33 | 0.29 | 58 | 6 | 10 | 12 | 2 | 0 | 0 | 3 | 0.68 | 136 | 3 | 4 | 1 | 3 | 3 | 5 | 14 |
| Dir. | 29 | 0.30 | 52 | 6 | 13 | 2 | 4 | 1 | 0 | 3 | 0.55 | 127 | 0 | 2 | 3 | 5 | 1 | 8 | 10 |

Scaff., Experimental group; Dir., Direct Instruction group; M, Average number of resolved out of total; Total, Cumulative

## Mathematical problem algorithm applied

A comparison of the means shows that the percentage of algorithms applied in the scaffolding group increases from 58 ($M = 0.29$) to 136 ($M = 0.68$) and for the direct instruction group increases from 52 ($M = 0.30$) to 127 ($M = 0.55$). Further examination of the data by means of cross-tabs analysis shows that both groups applied more algorithms in the post-test and used an algorithm to find the solution to all six mathematical problems. Table 2 shows the data when an algorithm was used while solving mathematical problems.

## Mathematical problem algorithm correctly constructed

A comparison of the means shows that the percentage of correctly constructed algorithms for the scaffolding group increases from 47 ($M = 0.24$) to 89 ($M = 0.45$) and for the direct instruction group increases from 41 ($M = 0.24$) to 82 ($M = 0.47$). Further examination of the data by means of cross-tab analysis shows that, comparing the pre-test with the post-test, the number of correctly constructed algorithms increases. It is noteworthy that both groups in the post-test not only constructed more correct algorithms but also solved more complex mathematical problems. Table 3 shows the data for the correct construction of an algorithm to solve the mathematical problem.

**Table 3** Mathematical problem algorithm correctly constructed

| | | Pre-test | | | | | | | | | Post-test | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Mathematical problem correct algorithm | | | | | | | | | Mathematical problem correct algorithm | | | | | | |
| Group | *n* | *M* | Total | 0 | 1 | 2 | 3 | 4 | 5 | 6 | *M* | Total | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Scaff. | 33 | 0.24 | 47 | 6 | 10 | 14 | 3 | 0 | 0 | 0 | 0.45 | 89 | 4 | 4 | 7 | 10 | 3 | 1 | 4 |
| Dir. | 29 | 0.24 | 41 | 7 | 12 | 4 | 4 | 1 | 1 | 0 | 0.47 | 82 | 0 | 4 | 7 | 13 | 1 | 3 | 1 |

Scaff., Experimental group; Dir., Direct instruction group; M, Average number of resolved out of total; Total, Cumulative

**Table 4** Influence of the instruction variant used on the quality of the problem solution and necessary runtime

| | Quality SRA (average) | | | Applied SRA (average) | | | Necessary runtime (average) | | |
|---|---|---|---|---|---|---|---|---|---|
| Group | *n* | *M* | *SD* | *n* | *M* | *SD* | *n* | *M* | *SD* |
| Scaff. | 33 | 0.73 | 0.49 | 33 | 0.45 | 0.26 | 33 | 29.41 | 12.59 |
| Dir. | 29 | 0.59 | 0.45 | 29 | 0.41 | 0.29 | 29 | 34.14 | 10.06 |

Scaff., Experimental group; Dir., direct instruction group; $p = 0.05$. View based on the specified variables in the data analysis section

## Degree of algorithmic skill by programing according to the SRA-cycle

Analysis of the dissolved grid diagrams and the number of applied and correctly constructed algorithms shows that, after the SRA robotics intervention was conducted, more grid diagrams were correctly solved (an increase from 187 to 199), more algorithms have been applied (increase from 110 to 263), and more correct algorithms have been constructed (increase from 88 to 171).

## Influence of the instruction variant on the quality of the problem solution and the runtime required

In order to make visible the influence of the instruction variant used, the quality of the solution to the programming problem and the runtime it takes the robot to complete the programme was examined. Therefore, three two-tailed t-tests are used with as independent variable the two instruction variants (direct instruction/scaffolding) and as the dependent variables: (1) Was SRA used? (2) If so, to what extent was SRA applied? (3) The average runtime required by the robot to execute the program. Table 4 shows the data for the influence of the instruction variant.

### Quality of problem solution

A comparison of the means makes visible that the scaffolding group ($M = 0.73$, $SD = 0.49$) shows a slightly higher quality level of the problem solution than the direct instruction group ($M = 0.59$, $SD = 0.45$). Analysis of the t-test shows that there is no significant difference. From this it can be assumed that instruction through scaffolding, when learning

SRA-programming does not lead to a qualitatively higher problem solution level. Since the value "0" is within the confidence interval, 95% CI [−0.08, 0.35], this confirms the assumption that scaffolding does not result in a significantly higher qualitative problem solution level.

### Amount of SRA

A comparison of the means makes visible that the scaffolding group ($M = 0.45$, $SD = 0.26$) applies SRA more than the direct instruction group ($M = 0.41$, $SD = 0.29$). Analysis of the t-test for the amount of SRA used shows that there is no significant difference. From this it can be assumed that instruction through scaffolding does not lead to an increase of applying SRA in a problem solving based programming environment. Since the value "0" is within the confidence interval, 95% CI [0.18, −0.81], this confirms the assumption that scaffolding does not significantly lead to more use of SRA.

### Runtime of program

A Comparison of the means makes visible that the robot programmed by the scaffolding group ($M = 29.41$, $SD = 12.59$) needs less runtime to execute the program than the robot programmed by the direct instruction group ($M = 34.14$, $SD = 10.06$). *T*-test analysis on the runtime the robot needs to execute the program shows that there is no significant difference. From this it can be assumed that instruction through scaffolding, when applying SRA in a problem based programming environment when the programmed robot needs to execute its task, does not decreases. Since the value "0" is within the confidence interval, 95% CI [−10.81, 1.16], this confirms the assumption that the instruction variant scaffolding does not provide a significantly more efficient program and was therefore not visible in a faster execution runtime of the robot.

### Degree of self-efficacy by programming according to the SRA-cycle

To determine the influence of SRA programming on self-efficacy, a paired t-test is used to compare pre- and post-test measurement. The mean of self-efficacy in the pre-test ($M = 2.95$, $SD = 0.42$) is lower than the mean in the post-test ($M = 3.00$, $SD = 0.46$). Analysis of the paired samples t-test shows that there is no significant difference. Since the value "0" is within the confidence interval, 95% CI [−0.21, 0.01], this confirms the assumption that SRA-programming does not provide a significantly higher level of self-efficacy.

### Applying SRA using Lego Robotics through scaffolding related to the ability of solving grid diagrams

To assess whether the use of SRA programming through scaffolding leads to a more successful level of solving grid diagrams i.e. more problems solved a one-tailed t-test is used. As an independent variable the instruction variant was used (scaffolding/direct instruction) and as a dependent variable the number of correctly dissolved grid diagrams (difference between pre- and post-test).

Analysis of the means makes visible (see: Table 1) that the scaffolding group, comparing the pre-test ($M = 0.50$, $SD = 0.17$) with the post-test ($M = 0.52$, $SD = 0.25$), shows a

slightly lower level of solving grid diagrams than the direct instruction group, which scored on the pre-test ($M = 0.51$, $SD = 0.19$) and on the post-test ($M = 0.55$, $SD = 0.22$), Analysis of the t-test shows that there is no significant difference between the two instruction groups. From this it can be assumed that an instruction according to scaffolding does not lead to a better resolution of grid diagrams in comparison with a direct instructional approach. Since the value "0" is within the confidence interval, 95% CI [−0.08, 0.15], this confirms the assumption that scaffolding does not lead to solve more grid diagrams. The hypothesis that learning to apply SRA-programming in a Lego robotics context when using the instruction method scaffolding leads to a more successful level of solving mathematical grid diagrams is rejected.

### Applying SRA with Lego Robotics through scaffolding related to the level of algorithmic skill

To assess whether the application of SRA-programming through scaffolding leads to higher level of algorithmic skill i.e. more complex diagrams solved two one-way t-tests are used. As an independent variable the two instruction variants are used and as a dependent variable(s) the applied algorithms (difference between post-test and pre-test) and the correct construction of algorithms (difference between pre-test and post-test) are used.

### Level of algorithmic skill by applying algorithms

Analysis of the means shows (see: Table 2) that the scaffolding group, in a comparison of the pre-test ($M = 0.29$, $SD = 0.27$) with the post-test ($M = 0.68$, $SD = 0.36$), used more algorithms than the direct instruction group which scored on the pre-test ($M = 0.30$, $SD = 0.30$) and on the post-test ($M = 0.55$, $SD = 0.28$).

Analysis of the t-test shows that there is no significant difference between the two instruction groups. From this it can be assumed that the used instruction variant scaffolding did not deploy more algorithms resulting in a higher level of algorithmic skill compared to the direct instruction group. Because the value "0" is within the confidence interval, 95% CI [−0.21, 0.11], it can't be claimed that the instructional variant scaffolding ensures a better solution of grid diagrams. The hypothesis cannot be ratified and is therefore rejected.

### Level of algorithmic skill by correctly constructing algorithms

Analysis of the means shows (see: Table 3) that the scaffolding group, in a comparison of the pre-test ($M = 0.24$, $SD = 0.15$) with the post-test ($M = 0.45$, $SD = 0.29$) constructs less correct algorithms then the direct instruction group which scored on the pre-test ($M = 0.24$, $SD = 0.22$) and on the post-test ($M = 0.47$, $SD = 0.21$).

Analysis of the t-test shows that there is no significant difference between the two instruction groups. It can be deduced from this that the instruction variant scaffolding did not result in a better level of constructing correct algorithms than in the direct instruction group. Therefore, as a consequence no higher algorithmic skill was shown, i.e. no more difficult grid diagrams were solved. Since the value "0" is within the confidence interval, 95% CI [0.11, −0.14], this does not lead to the assumption that the instruction variant scaffolding ensures better solution of grid diagrams. The hypothesis based on the application of algorithms cannot be ratified.

### Instruction trough scaffolding related to the level of self-efficacy compared to instruction through direct instruction

To assess whether instruction via scaffolding leads to a higher level of self-efficacy than through direct instruction a one-tailed t-test is used. As an independent variable the instruction variant was used (scaffolding/direct instruction) and as a dependent variable self-efficacy.

Analysis of the means makes visible that the scaffolding group ($M = 3.09$, $SD = 0.48$) scores marginally better on self-efficacy than the direct instruction group ($M = 3.00$, $SD = 0.45$). Analysis of the t-test shows that there is no significant difference between the two instruction groups. From this it can be assumed that the used instruction variant did not result in a higher level of self-efficacy. Since the value "0" is within the confidence interval, 95% CI [− 0.17, 0.30], this leads to the assumption that it can't be claimed that the instruction variant scaffolding leads to a higher level of self-efficacy. The hypothesis is rejected.

## Conclusions

An interpretation of the available data suggests that the robotic intervention, when using SRA-programming, has an effect on the respondents' mathematical skills. Comparing the pre-test with the post-test shows that pupils who program Lego robots with a SRA-approach apply more algorithms, construct more correct algorithms, and solve more difficult algorithms. Programming with the SRA-cycle seems to be the main cause of the increase of applied and correctly constructed algorithms. It can be assumed that programming with the SRA-cycle leads to a higher algorithmic skill and thus has a positive effect on the respondents' mathematical skills. The application and construction of algorithms when solving mathematical problems offered as grid diagrams even shows values that have more than doubled prior to the robotics intervention.

The hypothesis that learning to apply SRA-programming in a Lego robotics context when using the instruction method scaffolding leads to a more successful level (number) of solving mathematical grid diagrams can only be partially confirmed. The hypotheses that learning to apply SRA-programming in a Lego robotics context when using the instruction method scaffolding leads to a higher level of algorithmic skill (algorithm applied/correct algorithm constructed) can only be partially confirmed.

The hypothesis that SRA-programming leads to a higher degree of self-efficacy cannot be confirmed. Although in the literature a relationship has been found between programming and self-efficacy, the findings in this research show only a slight impact. It is not demonstrable that SRA-programming leads to a higher level of self-efficacy.

The hypothesis that using the instruction method scaffolding leads to a higher level of self-efficacy compared to the direct instruction method is not demonstrable. The difference in the influence of the applied instruction variant direct instruction versus scaffolding is not significant. However, there are clear indications (Table 4) that the instruction variant scaffolding remarkably lowers the execution time the robot requires to complete the challenge tasks.

## Discussion

This research aims to find answers whether SRA-programming using programmable robots in different types of instruction leads to an increase of algorithmic thinking and provides a higher level of self-efficacy.

Based on indications in the data it can be assumed that applying the SRA-cycle, when programming Lego robotics as an intervention, can create a higher level of algorithmic skill. This is in line with claims made by Cejka et al. (2006), Highfield et al. (2008), Kafai and Resnick (2012) and Silk et al. (2009) who state that solving ICT-issues can have a transfer to mathematical thinking and reasoning.

Our results show that the instruction variant (direct instruction/scaffolding) influences the quality of the created solutions and the execution time of a programmed robot. We also found some indications that the more open form (scaffolding) leads to improved outcomes. This calls into question the assertions of Kirschner et al. (2006) who claim that offering a fully directly guided instruction, compared to a more open form of guidance, provides sufficient understanding to achieve impact and greater learning efficiency. However, for the aspect of strategic and more effective learning, the indications we found are more in line with (Hmelo-Silver et al. 2007) who state that in a minimally guided instructional approach, cognitive load reduces and ensures that strategic and more effective learning and time effectiveness occurs. In a planned follow-up study, we expect to gain a better insight into the relevance of the indications that scaffolding generates better yields.

The hypothesis, that using the instruction method scaffolding leads to a significantly higher level of self-efficacy than using direct instruction, can't be shown. Programming with the SRA-cycle in a Lego robotics context, measuring self-efficacy comparing the pre-test and post-test of respondents from both instruction groups, shows marginally differences. This is remarkable, because the expected yield was that respondents, who received a minimally guided instruction by means of scaffolding in a more open setting, were given an excellent opportunity to reach a higher level of self-efficacy than respondents who received a strongly led instruction. This is not in line with the view of Bandura (1977) who states that being able to use one's own capacities in situations where the learner has more control would lead to a higher degree of self-efficacy. Furthermore, various studies have established that the short-term results of direct instruction and an instruction according to scaffolding are relatively similar (Kapur and Rummel 2012; Schmidt and Bjork 1992). However, the scaffolding approach usually leads to greater retention in the long term (Karaçalli and Korur 2014; Kvam 2000). Therefore, it would be worth investigating which instruction variant shows the most remaining returns over a longer period of time.

Apart from the stated research question, it is notable that, despite a focused instruction and a comprehensive exploration of parallel programming, respondents quickly fall back on simpler sequential programming. It is remarkable that the programming experience gained earlier, by exploring and solving the twenty programming tasks offered in the training and exercise sessions, is used superficially when programming the last five challenge assignments.

This research contributes to the theory of computational thinking and ICT education in primary schools. Robotic programming environments make pupils use computer technology to examine and solve problems systematically and in a creative and interactive way with sufficient possibilities for differentiation. SRA-programming in itself seems to be a not explicitly identified characteristic of computational thinking.

This research also contributes to the practical shaping of robotics education in classrooms. Despite the low level of significance, it can be concluded from the results that learning efficiency and the learning effect are present in both instructional variants. Therefore, it can be said that one could also match on instruction needs of respondents because the learning effect and learning return is primarily delivered by the intervention. This clears the way for a tailored choice of teacher's interventions which connect the principles of education, inquiry-based learning and adaptive learning. It makes possible to use mixed learning environments where pupils on the one hand can learn to program robots via a less guided, self-managing way, and on the other hand can profit from a more guided approach. However, the guided instructional approach has some limitations because the teacher doesn't always has time to guide pupils when needed. This pleads that pupils are as self-sufficient as possible. This is in line with the assertions of Slangen (2016) who argues that, when using robotics and programming, an exploratory free learning environment must be created in which the teacher has a less steering and more accompanying role.

## Limitations and follow up research

There are some considerations as to why our findings are below the required level of significance. Reasons may be that the programming assignments were too difficult, that pupils used the trial and error method to solve the grid diagrams or that there has been a development among pupils over time as a result of regular education.

This research is conducted within a Lego robotics-context. This context is chosen in order to activate SRA-thinking among primary school pupils. It can't be said with certainty that only SRA-programming using Lego robotics generates the best yields in research. To objectively compare as possible, and to be able to generalize from the obtained research results, it is desirable that the research is repeated with other ICT learning and programming environments.

From this research it is clear that the instruction variant has no significant effect on the learning outcomes. It would be worthwhile to investigate if and how teacher interventions in the different instructional variants affect learning of pupils. Further would it be desirable to investigate why pupils frequently recourse to sequential handling processes in programming and why they make little use of parallel flows despite that they have received a comprehensive instruction on this particular point. It would also be desirable to investigate if other ICT programming environments provide similar or perhaps higher learning revenue then the Lego robotics environment.

In this research, there was a limitation on the available working time with respondents. If more time had been available, this could have led to different results. It would also have been interesting to find out whether retention levels would differ on the basis of the two instruction variants used.

In this research the researcher was also the supportive teacher. It is important to take into consideration that teachers should be well-equipped for supporting this type of learning (Breed 2003; Slangen et al. 2011a). It asks for specific guidance skills and teacher competencies in the domain of ICT education and this closely follows the acquirement of twenty-first century skills in the field of computational thinking.

**Acknowledgements** This work was conducted while the first author was supported by a grant from the Dutch Organisation National Regieorgaan Praktijkgericht Onderzoek SIA (reference: NWO/Subsidie

KIEM.21V01.004). The authors would like to thank Heutink Netherlands for the donation of the required sets Lego Mindstorms and their cooperation.

## Compliance with ethical standards

**Conflict of interest** The authors declare that they have no conflict of interest.

**Ethical standards** The Ethical research board (cETO) of the Open University of the Netherlands has assessed the proposed study and concluded that this study is in line with the rules and regulations and the ethical codes for research in Human Subjects (reference: U2019/01324/SVW).

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

## References

Aesaert, K., Braak, Jv, Nijlen, Dv, & Vanderlinde, R. (2015). Primary school pupils' ICT competences: Extensive model and scale development. *Computers & Education, 81,* 18.

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review, 84*(2), 25.

Barry, P. (2006). On integer-sequence-based constructions of generalized Pascal Triangles. *Journal of Integer Sequences*, 9.

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, *38*(6), 20–23.

Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. In *Proceedings of the 42nd ACM technical symposium on computer science education* (pp. 245–250). ACM.

Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education, 58*(3), 978–988.

Bers, M. U., Ponte, I., Juelich, C., Viera, A., & Schenker, J. (2002). Teachers as designers: Integrating robotics in early childhood education. *Information Technology in Childhood Education Annual, 2002*(1), 123–145.

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist, 26*(3–4), 369–398.

Brave, F., Baardman, E., & Moussault, A. (2011). *Wegwijzer bij methoden voor leerprocessen*. Zaltbommel: Van Haren Publishing.

Breed, B. (2003). *The reflective abilities of expert and novice learners in computer programming.* Paper presented at the British Educational Research Association Annual Conference, Heriot-Watt University, Edinburgh, 11–13 September 2003.

Cejka, E., Rogers, C., & Portsmore, M. (2006). Kindergarten robotics: Using robotics to motivate math, science, and engineering literacy in elementary school. *International Journal of Engineering Education, 22*(4), 711.

Cito. (1987). Leerlingvolgsysteem Primair Onderwijs. https://www.cito.nl/onderwijs/primair-onderwijs.

Crasborn, F., & Hennissen, P. (2010). *The skilled mentor. Mentor teachers' use and acquisition of supervisory skills*. Eindhoven: Eindhoven School of Education.

Drijvers, P. (2015). Kernaspecten van wiskundig denken. *Euclides,* p. 6.

Drijvers, P., Kieran, C., Mariotti, M.-A., Ainley, J., Andresen, M., Chan, Y. C., et al. (2009). Integrating technology into mathematics education: Theoretical perspectives. In *Mathematics education and technology-rethinking the terrain* (pp. 89–132). Berlin: Springer.

Field, A. (2013). *Discovering statistics using IBM SPSS statistics*. Beverly Hills: Sage.

Futschek, G., & Moschitz, J. (2011). Learning algorithmic thinking with tangible objects eases transition to computer programming. In *International conference on informatics in schools: Situation, evolution, and perspectives* (pp. 155–164). Springer.

Guzdial, M. (2008). Education paving the way for computational thinking. *Communications of the ACM, 51*(8), 25–27.

Hasan, B. (2003). The influence of specific computer experiences on computer self-efficacy beliefs. *Computers in Human Behavior, 19*(4), 8. https://doi.org/10.1016/S0747-5632(02)00079-1.

Highfield, K., Mulligan, J., & Hedberg, J. (2008). Early mathematics learning through exploration with programmable toys. In *Proceedings of the joint meeting of PME* (Vol. 32, pp. 169–176). Citeseer.

Hmelo-Silver, C. E., Duncan, R. G., & Chinn, C. A. (2007). Scaffolding and achievement in problem-based and inquiry learning: A response to Kirschner, Sweller and Clark. *Educational Psychologist, 42*(2), 99–107.

Hogan, K., & Pressley, M. (1997). *Scaffolding student learning*. Cambridge, Massachusets: Brookline Books.

Horn, M. S., Solovey, E. T., Crouser, R. J., & Jacob, R. J. (2009). Comparing the use of tangible and graphical programming languages for informal science education. In *Proceedings of the SIGCHI conference on human factors in computing systems, Boston, MA, USA* (pp. 975–984). ACM.

Igbaria, M., & Iivari, J. (1995). The effects of self-efficacy on computer usage. *Omega, 23*(6), 587–605.

Jacobs, V. R., Lamb, L. L. C., & Philipp, R. A. (2010). Professional noticing of children's mathematical thinking. *Journal for Research in Mathematics Education, 41*(2), 33.

Jeschke, S., Kato, A., & Knipping, L. (2008). The engineers of tomorrow: Teaching robotics to primary school children. In *Proceedings of SEFI annual conference 2008*. Dansk Center for Ingeniøruddannelse.

Jonassen, D. H. (2006). *Modeling with technology: Mindtools for conceptual change*. Upper Saddle River, New Jersey: Pearson Merrill Prentice Hall.

Kafai, Y. B., & Resnick, M. (2012). Introduction. In *Constructionism in practice* (pp. 13–20). London: Routledge.

Kapur, M., & Bielaczyc, K. (2012). Designing for productive failure. *Journal of the Learning Sciences, 21*(1), 45–83.

Kapur, M., & Rummel, N. (2012). Productive failure in learning from generation and invention activities. *Instructional Science, 40*(4), 645–650.

Karaçalli, S., & Korur, F. (2014). The effects of project-based learning on students' academic achievement, attitude, and retention of knowledge: The subject of "electricity in our lives". *School Science and Mathematics, 114*(5), 224–235.

Kawalkar, A., & Vijapurkar, J. (2011). Scaffolding science talk: The role of teachers' questions in the inquiry classroom. *International Journal of Science Education, 35*(12), 43.

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*(4), 245–255.

Kennisnet. (2015). *Computing-onderwijs in de praktijk - Wat kunnen we leren van de Britten?* (p. 81). Zoetermeer: Kennisnet.

King, S. O., Stein, M. K., & Schunn, C. (2012). Designing educative guides: Reconceptualizing teacher's role in teacherless cognitive tutor-based robotics instruction. *American Educational Research Association*, 19.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 41*(2), 14.

Kolovou, A., Heuvel-Panhuizen, M. V. d., Bakker, A., & Elia, I. (2008). An ICT environment to assess and support students' mathematical problem-solving performance in non-routine puzzle-like word problems. In *Research in mathematics education* (p. 15). Nicosia, Cyprus: University of Cyprus.

Kong, S. C. (2008). A curriculum framework for implementing information technology in school education to foster information literacy. *Computers & Education, 51*(1), 129–141.

Korkmaz, Ö. (2018). The effect of scratch-and lego mindstorms Ev3-Based programming activities on academic achievement, problem-solving skills and logical-mathematical thinking skills of students. *Malaysian Online Journal of Educational Sciences, 4*(3), 73–88.

Korzilius, H. (2000). *De kern van survey-onderzoek*. Assen: Van Gorcum.

Krumholz, N. (1998). *Techno-logic: A micro-world for constructivist science and technology learning.* Paper presented at the ICCE98, Beijing, China.

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research, 2*(4), 429–458.

Kvam, P. H. (2000). The effect of active learning methods on student retention in engineering statistics. *The American Statistician, 54*(2), 136–140.

Leenders, Y., Naafs, F., & Oord, Ivd. (2010). *Effectieve instructie. Leren lesgeven met het activerende, directe instructiemodel*. Amersfoort: CPS.

Lindh, J., & Holgersson, T. (2007). Does lego training stimulate pupils' ability to solve logical problems? *Computers & Education, 49*(4), 1097–1111.

Liu, E. Z. F., Lin, C. H., & Chang, C. S. (2010). Student satisfaction and self-efficacy in a cooperative robotics course. *Social Behavior and Personality, 38*(8), 1135–1146.

Maas, P. (2015). *CodeKlas. Waarom we kinderen zouden leren programmeren*. Groningen: BoekTweePuntNul.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*(2), 137–168.

Pelgrum, W. J. (2001). Obstacles to the integration of ICT in education: Results from a worldwide educational assessment. *Computers & Education, 37*(2), 163–178.

Ploeg, V. d. (2015). Efficient abstractions for visualization and interaction. In U. v. Amsterdam (Ed.). Amsterdam, Netherlands.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004). *Self-Efficacy and Mental Models in Learning to Program.* Paper presented at the Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, Leeds, UK.

Ramalingam, V., & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research, 19*(4), 14.

Santos, J. R. A. (1999). Cronbach's alpha: A tool for assessing the reliability of scales. *Journal of Extension, 37*(2), 1–5.

Sapounidis, T., Demetriadis, S., & Stamelos, I. (2015). Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing, 19*(1), 225–237.

Sargent, R., Resnick, M., Martin, F., & Silverman, B. (1996). Building and learning with programmable bricks. In Y. Kafai & M. Resnick (Eds.), *Constructionism in practice; designing, thinking, and learning in a digital world* (pp. 161–173). Mahwah, New Jersey: Lawrence Erlbaum Associates, Publishers.

Schmidt, R. A., & Bjork, R. A. (1992). New conceptualizations of practice: Common principles in three paradigms suggest new concepts for training. *Psychological Science, 3*(4), 207–218.

Schoenfeld, A. H., & Sloane, A. H. (2016). *Mathematical thinking and problem solving* (Vol. 1). New York: Routledge.

Scholz, U., Doña, B. G., Sud, S., & Schwarzer, R. (2002). Is general self-efficacy a universal construct? Psychometric findings from 25 countries. *European Journal of Psychological Assessment, 18*(3), 242.

Schwarzer, R., & Jerusalem, M. (1995). Generalized Self-Efficacy scale. *Measures in Health Psychology: A User's Portfolio. Causal and Control Beliefs, 1,* 35–37.

Silk, E., Higashi, R., Shoop, R., & Schunn, C. (2010). Designing technology activities that teach mathematics. *The Technology Teacher, 69*(4), 21–27.

Silk, E., & Schunn, C. (2008). *Using Robotics to Teach Mathematics*. Paper presented at the American Society for Engineering Education Annual Conference 2007, Pittsburgh, PA.

Silk, E., Schunn, C., & Shoop, R. (2009). Motivating efficiency & meaning in problem solving with robotics. *Carnegie Mellon Robotics Academy*.

Slangen, L. (2016). *Teaching robotics in primary school*. Eindhoven: Eindhoven University of Technology.

Slangen, L., Fanchamps, N., & Kommers, P. (2008). A case study about supporting the development of thinking by means of ICT and concretisation tools. *International Journal of Continuing Engineering Education and Life-Long Learning, 18*(3), 18.

Slangen, L., Keulen, H. v., & Gravemeijer, K. (2011a). Preparing teachers to teach robotics in primary schools. In *Professional development for primary teachers in science and technology* (pp. 181–198). Berlin: Springer.

Slangen, L., Keulen, H. v., & Gravemeijer, K. (2011b). What pupils can learn from working with robotic direct manipulation environments. *International Journal of Technology and Design Education, 21*(4), 20.

Suárez, Á., Specht, M., Prinsen, F., Kalz, M., & Ternier, S. (2018). A review of the types of mobile activities in mobile inquiry-based learning. *Computers & Education, 118,* 38–55.
Tay, L. Y., Lim, C. P., Nair, S. S., & Lim, S. K. (2014). Online software applications for learning: Observations from an elementary school. *Educational Media International, 51*(2), 15.
Teeuw, B., Schwarzer, R., & Jerusalem, M. (1994). Dutch adaptation of the general self-efficacy scale. Berlin.
Thijs, A., Fisser, P., & Hoeven, M. v. d. (2014). Digitale geletterdheid en 21e eeuwse vaardigheden in het funderend onderwijs: Een conceptueel kader.
Toh, L. P. E., Causo, A., Tzuo, P.-W., Chen, I.-M., & Yeo, S. H. (2016). A review on the use of robots in education and young children. *Journal of Educational Technology & Society, 19*(2), 148–163.
Veenman, S. (2001). Directe Instructie. (Vol. 1, pp. 26). Nijmegen, Netherlands: Katholieke Universiteit Nijmegen.
Voogt, J., & Roblin, N. P. (2010). *21st century skills*. Enschede: Discussienota.
Vosniadou, S., Ioannides, C., Dimitrakopoulou, A., & Papademetriou, E. (2001). Designing learning environments to promote conceptual change in science. *Learning and Instruction, 11*(4–5), 381–419.
Wang, D., Wang, T., & Liu, Z. (2014). A tangible programming tool for children to cultivate computational thinking. *The Scientific World Journal, 2014,* 10.
Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.
Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th international conference on interaction design and children* (pp. 199–208). ACM.
Wilensky, U. (1995). Making sense of probability through paradox and programming: A case study in a connected mathematics framework. *Journal of Mathematical Behavior*, *14*(2), 26.
Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 3.
Wong, L. L. (2014). Rethinking the sense-plan-act abstraction: A model attention and selection framework for task-relevant estimation. In *Workshops at the twenty-eighth AAAI conference on artificial intelligence, Quebec, Canada.*
Wyeth, P., Venz, M., & Wyeth, G. (2004) Scaffolding Children's Robot Building and Programming Activities. In: Polani D., Browning B., Bonarini A., Yoshida K. (eds) RoboCup 2003: Robot Soccer World Cup VII. RoboCup 2003. *Lecture Notes in Computer Science*, vol 3020/2004 (pp. 308–319). Springer, Berlin, Heidelberg.
Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends, 60,* 4.
Zimmerman, B. J. (2000). Self-efficacy: An essential motive to learn. *Contemporary Educational Psychology, 25,* 10.

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.