The Effect on Computational Thinking Using SRA-Programming Anticipating Changes in a Dynamic Problem Environment

Fanchamps, Nardie; Slangen, Lou; Specht, Marcus M.; Hennissen, Paul

DOI
10.1109/TLT.2022.3162895

Publication date
2022

Document Version
Final published version

Published in
IEEE Transactions on Learning Technologies

The Effect on Computational Thinking Using SRA-Programming: Anticipating Changes in a Dynamic Problem Environment

Nardie Fanchamps, Lou Slangen, Marcus Specht, and Paul Hennissen

***Abstract*—This article illustrates that the task design and problem selection are of characteristic influence to evoke sense-reason-act programming (SRA) among primary school pupils when programming robots. Research shows that the task design influences the development of computational thinking (CT). The literature provides evidence that the context, the problem space, and the representation of the problem to apply SRA-programming require the programming task to be embedded in a dynamic context in which a programmable robot must use sensory information to anticipate changes in the environment. In order to ascertain whether the problem space and the task design influence the evocation of SRA-thinking, this article uses a research design comparing the differences between two programming conditions (static/dynamic). In these conditions, pupils use Lego EV-3 robots and Mindstorms software to solve programming problems. As a post-measurement, a Lego challenge is applied. In this article, it is shown that the integration of a dynamic task design to solve a programming problem is essential for a deeper understanding of CT skills. Furthermore, when pupils can immediately test the consequences of their program in a dynamic environment and, thus, the learning environment provides an appropriate problem, they gain a deeper understanding of the added value of sensors and will be better able to reason about complex problems. It is found that programming in a dynamic problem environment almost naturally evokes SRA-thinking, as opposed to programming in a static environment. The influence of SRA-programming as demonstrated identifies characteristics of CT.**

***Index Terms*—Computational thinking (CT), dynamic environments, robotics, sense-reason-act (SRA)-thinking, tangible output.**

I. Introduction

Learning to program is becoming increasingly prominent in primary education [1]. Primary education is the designated place to teach pupils at an early stage about programming and to teach them about its functionality and applicability [2]. Programming can be considered as a basic skill that should be integrated into the curriculum of primary schools to equip pupils with the necessary competences [3], [4].

Programming appears to make an important contribution for pupils to learn computational thinking (CT) [5], [6]. CT is the ability to describe complex problems using the basic concepts of computer algorithms [7].

Programmable robots offer an excellent opportunity to develop CT skills, because on the basis of a visually perceptible output, the result of the programming intervention is concrete and tangible [8]–[10]. If pupils can directly test the effect of their programming actions in reality, they will be better able to critically examine and assess their programming actions [11]. Because programmable robots can be used for gaining instant feedback on the consequences of code, they function as Direct Manipulation Environments (DME's) [12], [13].

DME's involve pupils in constructing mental models of phenomena. Pupils are challenged to directly manipulate parameters and variables in the environment. Many DME's strengthen the feeling of working with concrete objects. DME's make pupils through active participation reason, predict and hypothesize, analyze and test [11], [12]. Robots are concrete and physical DME's and can be controlled by programming making use of actuators and sensors [12], [13]. DME's have the capacity to provoke a learning dialog and higher order thinking skills (e.g., analyzing, synthesizing, evaluating, and causal reasoning), can provide a rich context for practicing and learning programming, and for developing (general) problem-solving skills (CT) in the context of robot programming [11].

Robots can be programmed to interact with their physical environment. To make this interaction possible, sense-reason-act (SRA) programming is applied. Previous research has provided indications that primary school pupils are able to reach a certain level of SRA-programming, but that pupils do not apply SRA as a matter of course [10], [14]. This is despite the fact that they have experienced the benefits of applying an SRA-approach in solving programming problems previously. Robots operating in an unchanging environment can often rely on linear, sequential programming depending on the task at hand. Robots that can anticipate changing conditions need sensor input and, as a consequence, results in programming based on principles of conditional reasoning. It is by using SRA-programming that a programmable robot can anticipate

Manuscript received July 18, 2020; revised February 19, 2021 and March 1, 2022; accepted March 22, 2022. Date of publication March 29, 2022; date of current version June 20, 2022. The work of Nardie Fanchamps was supported by Dutch Organization Nationaal Regieorgaan Praktijkgericht Onderzoek SIA under Grant NWO/Subsidie KIEM.21V01.051. *(Corresponding author: Nardie Fanchamps.)*

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by The Ethical research board (cETO) of the Open University of The Netherlands has assessed the proposed study and concluded that this article is in line with the rules and regulations and the ethical codes for research in Human Subjects (reference: U2019/01324/SVW).

Nardie Fanchamps is with Open Universiteit Nederland, 6419 AT Heerlen, The Netherlands (e-mail: nardie.fanchamps@ou.nl).

Lou Slangen is with the Fontys University of Applied Science, 6131 AJ Sittard, The Netherlands (e-mail: l.slangen@fontys.nl).

Marcus Specht is with the Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: m.m.specht@tudelft.nl).

Paul Hennissen is with the Zuyd University of Applied Science, 6419 DJ Heerlen, The Netherlands (e-mail: paul.hennissen@zuyd.nl).

Digital Object Identifier 10.1109/TLT.2022.3162895

changes in its surroundings [10]. SRA-programming activates the application of problem-solving solution strategies, which benefits the development of CT [14].

For further clarification, it seems necessary to make a specific distinction between SRA-thinking and SRA-programming. SRA-thinking refers to the logical reasoning process in which thinking in parallel structures and operating principles is the underlying rationale. It includes logical, causal, conditional, and iterative reasoning and the ability to make cause-and-effect relationships when specifically applying sensor input to anticipate changes in task design. SRA-thinking involves thinking in parameters and variables and calls upon higher-order thinking skills in reasoning, analyzing, synthesizing, and judging. SRA-thinking can be seen as a mental disposition [15] that leads to the initiation of the process of SRA-programming, and is distinct from SRA-programming which is based on the pragmatic use of coding tools. SRA-programming refers to programming that uses sensor-based input combined with parallel programming routines in which more complex programming concepts such as nested loops, conditionals if-then-else, and functions come into play. Therefore, this article examines whether the type of programming problem and the task design influence the evocation of SRA-thinking. It is investigated whether there is a characteristic and qualitative difference in the application of SRA-programming in a programming environment with either a dynamic or a static task design when primary school's pupils program Lego Robotics EV-3.

II. Theoretical Framework

In this article, we want to know whether a dynamic task design elicits SRA-thinking and enhances the development of CT. CT is a problem-solving approach in which complex, abstract problems are translated and reformulated in such a way that they can be solved using the fundamental concepts of computer science [7], [16].

CT is a conceptual way of thinking and includes processes like problem formulation, data organization, analysis, and representation to solve problems [17]. It encompasses a set of problem-solving skills, such as problem decomposition, algorithmic thinking, pattern recognition, debugging, parallelization, and abstraction [16], [18], [19]. CT stimulates the ability to define a distinct and structured sequence of basic and well-specified steps to solve complex problems [20]. The essence of CT involves dividing complex problems into more familiar/manageable subproblems (problem decomposition), using a sequence of steps (algorithms) to solve problems, reviewing how the solution transfers to similar problems (abstraction), and finally determining if a computer can be used efficiently to solve those problems (automation) [21], [22]. CT can be used explicitly to gain new insights into problems outside computer science, to establish relationships toward other fields, and can become an integrated part of education. It covers a wide range of computing principles, approaches, knowledge, and skills essential for solving problems that require a lot of information, variables, and computational power [23].

CT requires the application of strategies to solve problems. Costa [24] states that solving problems implies the appropriate application of knowledge in a specific situation that is strongly related to the content. The specific context in which learning occurs determines what types of actions are suitable and what impact those actions have [25]. Problem solving uses often an inquiry-based approach. Since the problem space of pupils contains only partial knowledge, it is necessary to initiate a general search for information, as well as to discover a state that contains a solution to the problem. This is to make pupils aware of their learning skills, their achievements, and their ability to reflect on what they have learned [26]. Pupils need to be aware of how to solve problems and must learn to recognize that their actions clearly put them in a strong position to learn from mistakes or successes [27]. A well-defined problem space and solution paths are usefully combined with strategies for finding suitable possibilities to solve a problem [28]. Solving the problem is not simply a search through a predetermined problem space. It is also a search for an appropriate representation of the problem [29]. The type of the problem definition seems to be of prominent importance in relation to finding a solution to the problem. The ability to solve problems evolves when developing strategies. Moreover, higher forms of problem solving are accomplished using the same basic routines that evolved for that purpose [27], [30], [31]. Learning the required problem-solving strategies and skills is what CT is associated with. In this respect, CT can not only influence the problem-solving skills of pupils in general, but also has an important influence on their generic development [32].

Characteristic for effectively solving a problem is a multistage process consisting of several phases to be appointed:

1) identifying a problem;
2) setting goals;
3) using solution strategies;
4) arriving at different directions of solution;
5) making reasoned decisions to arrive at the solution;
6) determining through evaluation;
7) reflection and introspection whether the problem is solved and/or the problem solution needs to be adjusted;
8) generalizing so that applicability in other situations is possible [31], [33].

Characteristic of a multiperspective, imaginative problem-solving process is that it is not linear and does not include strict, predetermined rules [34]. The solution process is usually unpredictable, iterative, or dynamic in nature and involves several verification moments [35].

Applying complex problems as a catalyst to promote learning of concepts and principles by learners in a learning environment can be regarded as problem-based learning (PBL) [34]. In order to encourage the transfer of knowledge and skills enabling pupils to use existing knowledge and to acquire new knowledge, this requires challenging, provocative problems that motivate to learn [27]. The application of robotics in primary school education can be a useful tool to relate problem-solving strategies to realistically occurring problem contexts [36]. Being able to reflect on problem-solving strategies through programming robots is important because the associated thought processes, decisions, and actions can determine the final learning result [37]. Robotics programming problems have a wide variety of solutions, but many can be improved in efficiency when placed in meaningful

contexts [18]. Solving problems by means of programming robots can have a positive influence on the quality of the programming solution itself and on the conceptual understanding [38]. Programming problem environments that are challenging in nature can make a greater contribution to conceptual understanding than simple, more orderly problem environments [18], [37].

From research, we know that primary school pupils mainly use linear programming structures when programming robots, despite the fact that they have learned the advantages of parallel programming through previous experiences [14]. In addition, pupils have also found it difficult to apply SRA, although they have already explored the benefits of programming with the SRA-approach [11]. The essence seems to be that pupils have difficulty using complex elements of SRA-programming, such as the "if-then-else," "wait until," or the "nested loop," which presuppose conditional reasoning [39]. From earlier research, we know that pupils' state of mind often tends to choose the easiest and most obvious solution rather than the most efficient one [14]. Comprehending SRA implies that pupils can establish the explicit relationship between processes in which a programmable robot:

1) registers external values of observations based on sensor use (sense → referring to sensor reading;
2) then compares these values of external observations with internal preset values which form conditionals by which the path to follow (reason → referring to decision making) is decided;
3) the activation of a subsequent process in which the program "informs" the robot on how the robot should act (act → referring to program response) [10], [39], [40].

These insights enable pupils to program a robot that, through the use of sensors and its program, can anticipate changes in its environment. In other words, pupils apply SRA-thinking through understanding the interactions and interdependencies between the program, sensors, and actuators used [11].

Using sensory input to solve robotic programming tasks can be considered as an effective way of programming [41]. As we argued before, SRA-programming is characterized by a first step in which input is obtained through sensory detection (*sense*). In a second step, the decision part (*reason*), the perception is measured against set conditions. Therefore, the computer program to be designed must anticipate what can be observed [11] and what the conditions are. As long as pupils do not yet understand that SRA-programming offers a more efficient and sometimes the only feasible alternative and can use linear programming structures to solve a programming task, they will not enter the mental disposition of SRA-thinking [14], [24], [42]. When the environment in which a programmable robot has to perform its tasks can vary continuously, and is therefore dynamic or unpredictable in nature and enforces the most efficient solution, then linear programing is no longer sufficient. This triggers a mental process in which the state of mind will play a role [15]. In order to evoke SRA-thinking, pupils need to call upon their disposition for ingenuity, originality, and insightfulness (e.g., applying past knowledge to new situations, striving for accuracy, thinking flexibly, thinking about thinking, questioning, and posing problems) to express SRA-thinking [43]. This requires a mindset in which the initial approach to solve a dynamic programming task is characterized by means of proactive programming, applying iterations, conditionals, and functions to successfully solve the programming problem [44], [45].

SRA requires a certain level of abstract thinking. It means being capable of analyzing the robotic task environment, i.e., being capable of recognizing the prerequisites and iterative conditions and translating them into the correct application of the programming instructions to be constructed [39], [46]. For instance, the selection for the type of sensor to apply and the range of values of a variable. When pupils understand that the reasoning process underlying programming is based on principles of logic, conditional, causal, and iterative reasoning, including thinking in parameters and variables, this should be recognizable as such in the computer programs they create [47]. Programming code produced according to SRA-thinking contains such structures and principles of programming. Analyzing pupils' constructed code, therefore, can provide information about pupils' level of application of SRA-thinking by means of the measured values for CT.

In order to enable problem solving through an application of technology in an effective and easily accessible way, the use of Lego Robotics in primary schools turns out to be an efficient manner for problem-solving activities [47], [48]. Results from various studies indicate that activities with programmable, tangible robots assisted pupils to reflect on the problem-solving decisions they made [10], [49], [50]. These studies also highlighted that pupils were able to relate their problem-solving strategies to real-world contexts [51]. Also is demonstrated that programming LEGO robots can be considered as useful problem-solving tool in the classroom to enhance CT [14]. This is to ascertain what problem-solving strategies primary school pupils use when working with LEGO robots and whether they are able to effectively relate their problem-solving strategies to real-world contexts [18]. Moreover, programmable robots as educational tools engage pupils in their own learning through active constructionist environments, which stimulate the development of higher order thinking and problem-solving skills, promoting pupils' conceptualization in meaningful authentic ways [36].

If pupils program tangible robots, two main problem orientations can be distinguished: unchanging, static conditions and changing, dynamic conditions [44], [52]. In a static problem space, the environmental conditions are not changing. The tasks to be accomplished by the robot are predefined and the program that has to provide the robot with information can be clearly defined in advance [53]. The static task environment is characterized by a clear, transparent design. The assignments to be performed for which a robot must be programmed are known in advance and are predictable because the task environment does not change. These nonchanging tasks can be solved both by linear programming and also by using sensors [54]. However, dynamic environments are characterized as unpredictable, the problem space is subject to continuous change and is not clear-cut in advance [45]. A dynamic environment includes a multitude of factors that must be anticipated on an ad-hoc basis [55], [56]. The subsequent actions to be taken by a programmable robot cannot be predicted in advance, the task that the robot has to accomplish is subject to change and the program must anticipate such changing conditions [57].

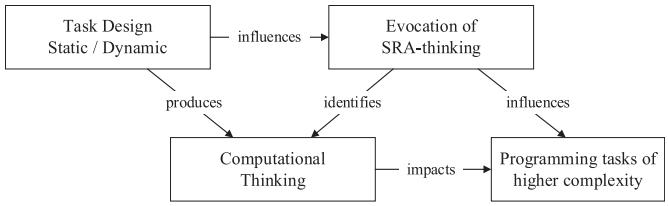

Fig. 1. Schematic representation of the conceptual model.

Building on the theoretical exploration above, we presume a correlation between the following:

1) the task design which is expected to produce development of CT caused by the nature of the problem (static/dynamic);
2) the evocation of SRA-thinking, e.g., more generic the influence on CT caused by differences in the task design;
3) the identification of CT emerged from the evocation of SRA-thinking;
4) the influence of the evocation of SRA-thinking on programming tasks of higher complexity;
5) the impact of CT on the construction of programming tasks of higher complexity.

Therefore, our conceptual model in Fig. 1 gives an overview of supposed relationships between independent and dependent variables that need to be further investigated.

III. Research Question, Subquestions, and Hypothesis

From previous research, we know that pupils revert to linear, sequential programming robots, despite having gained a previous learning experience exploring the benefits of the more efficient SRA-programming [10], [14]. The question that emerges is why pupils do not apply SRA-programming, while understanding and mastering fundamental principles and advantages underneath. We assume this seems to be attributable to a static task design with a predictable overview in which pupils are not confronted with unforeseen, changing events. However, if the programming environment in which a programmable robot has to perform its tasks varies and is, therefore, dynamic, unpredictable in nature enforcing the most efficient solution, the use of sensors, iterations, conditionals and functions, characteristic of SRA-programming is a necessity. To investigate this claim, we used a dynamic task design (simultaneous to static task design) that stimulates the application of SRA-programming where there is no possibility to find the programming solution through linear, sequential thinking. We conjecture that the nature of the problem space and the task design will cause pupils do or do not use SRA-programming in which the application of SRA identifies characteristics of CT. This assumption leads to the main research question: "What is the influence of the problem environment and the task design on evoking SRA-thinking when programming robots?"

In addition to the main research question, the following subquestions have been formulated.

1) What kind of problems require SRA-thinking and is SRA-thinking necessary to solve programming problems of higher complexity?
2) How can we design a scaffold toward more complex tasks to guide pupils toward SRA-thinking?
3) What is the impact of the programming environment on developing SRA-solutions?

In addition to the subquestions, three hypotheses have been formulated.

1) Pupils who solve programming assignments in a dynamic task design are able to activate SRA-thinking outflowing in working solutions.
2) Pupils who solve programming assignments in a dynamic task design apply SRA-thinking more than pupils who solve programming assignments in a static task design.
3) Pupils who solve programming assignments in a dynamic task design show problem solving of a qualitatively higher level than pupils who solve programming assignments in a static task design.

IV. Method

In order to investigate whether there is an influence of the programming problem given and the task design to evoke SRA-thinking, we have designed a programming environment with a two-pronged task setup. This setup was chosen as such because we wanted to compare whether there is a measurable difference in making use and understanding of SRA-thinking between the group that had to solve tasks in a static programming environment and the group that solved tasks in a dynamic programming environment. Analysis of the constructed programs from respondents and the information obtained from the observations makes it possible to find answers to our research questions, subquestions, and hypotheses.

To investigate the research questions and hypotheses, we have used a research design that comprises both qualitative and quantitative aspects, as shown in Fig. 2. This includes

1) a basic instruction on how to program using Lego Robotics EV-3;
2) a robotics intervention designed for training tasks;
3) a final programming challenge consisting of a static and dynamic task design.

The static, predictable task environment in our research is created to use fixed objects that need to be avoided. The dynamic environment in our research is created by constantly randomly moving objects, creating an unpredictable task environment.

In the final programming challenge, by means of observation of the execution and analysis of the constructed programs, we collected qualitative data that were then converted into quantifiable units to determine the frequency of the occurrence of a value. To be able to determine whether SRA had been used in both programming environments and to give value to the quality of the programming solution, we measured the characteristics

1) if pupils are able to successfully complete the programming tasks;
2) whether SRA was used;
3) we assessed the efficiency and quality level of the program that was constructed.

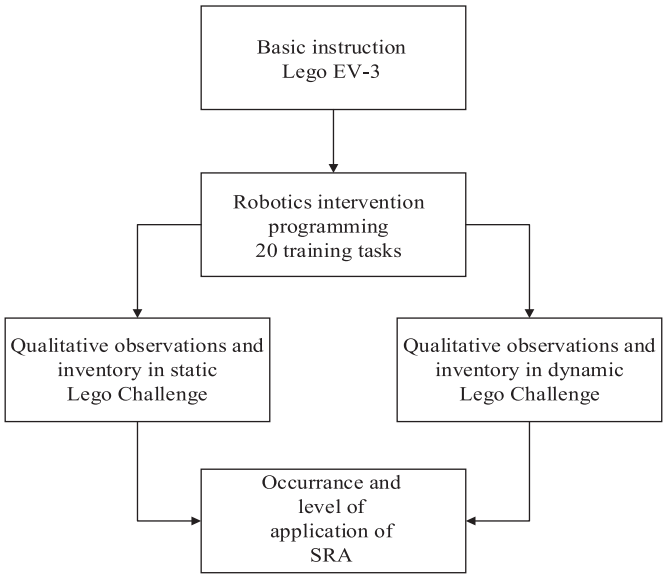

Fig. 2. Research design.

A. Participants

This research was conducted among pupils from grade 5[1] who are 10 and 11 years of age ($N = 29$) of a primary school in The Netherlands. Pupils are not familiar with programming, apart from the use of basic computer programs such as Word, PowerPoint, and Internet. The pupils were divided randomly into 14 subgroups consisting of 13 pairs and one triplet. Fig. 3 displays the division of participating groups.

After all subgroups had followed a basic training in how to program with Lego Mindstorms Robotics EV-3, each of the subgroups conducted 9 weeks, 1-h sessions to solve 20 programming tasks to demonstrate the solution devised. After completion of the 20 programming tasks, all subgroups had to solve five new challenging programming tasks. The group that had to solve these five challenges in the static environment could solve them by using linear, sequential programming, and/or by making use of SRA. The group that had to solve these five challenges in the dynamic environment could only successfully solve them by explicitly using SRA. We used this set-up as such to compare whether the tasks design and problem environment influence the evocation of SRA and to what level the use of SRA did or did not occur.

B. Materials

From previous research, we found indications that primary school pupils can use Lego EV-3 Mindstorms robots as a programming environment in a functional way [10], [14]. These programmable robots are controlled via a visually oriented programming environment in which the user has to connect definable blocks, containing programming commands, by dragging and sequencing them in the correct order on the worksheet [58]. Furthermore, these definable blocks consist of controllable parameters, variables, logical operators, et cetera

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

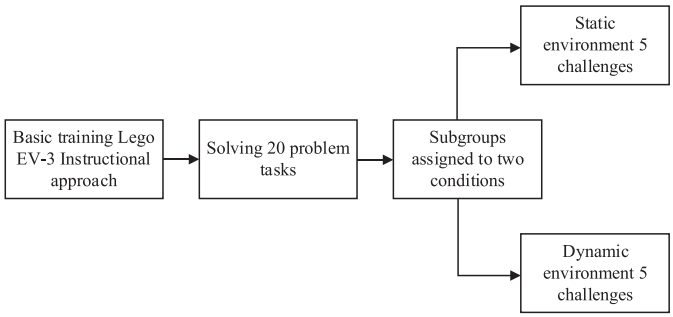

Fig. 3. Schematic representation of the group division.

which can also be influenced [9]. By manipulating the variables and sequencing the blocks in a specific order, pupils construct their program. In order to determine the differences in programming skills and the evocation of SRA-programming after the training sessions, a Lego robotics challenge is organized in which the influence of the problem environment and the task design occurred.

To demonstrate which programming environment and task design evokes SRA-thinking, we set up a predefined problem space in which pupils solved a Lego robotic programming challenge task. This problem space consists of two different playing fields in which pupils had to execute programming tasks making use of a predefined robot. This robot was equipped with push-button sensors at the front and back and a bumper bracket that allowed the robot to collide and rotate like a bumper car.

Playing field one, as shown in Fig. 4, contained fixed, static objects that the robot had to maneuver between to move from start to finish. More and more objects were added to increase the difficulty and challenge of the programming tasks. The assignments could still be solved with linear, sequential programming where pupils could decide for themselves whether to program with or without the more efficient use of SRA-programming and/or by application of push-button sensors.

On playing field two, as shown in Fig. 5, a dynamic environment has been created in which a programmed robot moves unpredictably around. The assignment was to let the self-programmed robot move on the playing field without getting stuck. To make the programming tasks increasingly difficult, and to provoke SRA-programming, a second comparable robot that moves unpredictably around was brought into the playing field. Furthermore, two other pairs of pupils were also allowed to add their robots into the playing field (with a maximum of 3), taking into account the fact that all five robots had to keep on moving/not get stuck. In order to do this, pupils' robots must be programmed making use of push-button sensors in such a way that they could react to each other and to anticipate unforeseen situations. This challenging assignment could absolutely not be solved by making use of linear, sequential programming, but only by making use of SRA-programming.

C. Procedure

All 14 subgroups received an identical introductory instruction, which also covered the application of sensors, on how to

Fig. 4. Static task design with fixed objects.

Fig. 5. Dynamic task design with multiple randomly moving robots.

use the Lego programming environment. Pupils learn in this basic training, by means of a "show-and-tell approach" and by using simple tasks, how the robot should be programmed to drive forward/backward, how to make a turn, how to move from A to B between obstacles without getting stuck, and how to apply the basic elements of SRA-programming (iterations, responding to changes in the task environment, functions). It was also demonstrated how to program by means of sensoric input, how the push-button sensor is operated, how the robot could respond by its program when using sensor activation, and how the use of sensors, by applying SRA-programming, can ensure a much smarter programming solution.

After the basic instruction, each of the subgroups conducted nine 1-h sessions to practice 20 training tasks so that SRA-programming can provide a more efficient and smarter way to solve robotics programming assignments. These tasks are characterized by a problem-solving approach and can be solved either using linear, sequential programming, or using SRA-programming.

In the final challenge assignments, all subgroups are randomly assigned to two different robotic problem-solving contexts (static/dynamic). The assignments in these challenges offer pupils a choice of whether or not to apply SRA-programming. To do so, pupils have all the options available and the robots to be operated are equipped with two push-button sensors to apply SRA-programming. It is precisely in the two different challenge conditions that it is expected to emerge that, although pupils understand and have learned the basic principles of the SRA-approach, they do not apply SRA-programming as a matter of course but that the application of SRA-programming is evoked by the difference (static/dynamic) in the task design. In this final static or dynamic robot programming challenge, pupils can show what they have learned from the 20 programming tasks and what occurs by using or not using SRA-programming. In this challenge task, five new programming tasks of increasing difficulty had to be solved. These are static (not changing) in nature for one group and dynamic (changing) in nature for the other group. In both groups, pupils could decide for themselves whether to program with or without the fully functional and most effective use of sensors. By observation during the programming sessions, by analysis of the constructed programs, and by inventory, the researcher kept track of (a) whether the programming task had been solved correctly (yes/no) and (b) whether SRA had been used (yes/no). From these observations, we determined how often each value occurred. Furthermore, we assessed the efficiency and quality level of the constructed programs as a qualitative indicator. To do this, we (c) recorded the fastest runtime out of three attempts as a predictive value to illustrate the level of quality and efficiency of the constructed program and (d) we analyzed the efficiency of the program that was constructed. In order to sufficiently benchmark this qualitative interpretation, the fourth indicator has been calibrated by a research group of eight education experts. For the purpose of retrospective analysis, the programs constructed by pupils were stored. The researcher used the data of the retrospective analysis to assess whether SRA had been applied and could give a value judgment on how efficient the constructed programs were.

V. Results and Data-Analysis

The main research question, "What is the influence of the programming environment and the task design on evoking SRA-thinking when programming robots?," is answered by analysis of the means for the dichotomous variables combined with an analysis of observed, qualitative data that are converted into quantifiable units through inventory. T-test analysis is used to investigate subquestions and to confirm or reject hypotheses. Observations have been converted into quantifiable data by means of systematic and structured scoring to enable quantitative analysis. The data derived from the Lego robotics challenge were entered into SPSS for quantitative data analysis. Other qualitative observations concerning the effect of using or not using SRA are included in this article as a descriptive comparison and result. The effect of the independent variables on the dependent variables is examined (see Fig. 1). Differences in values are determined by comparing the means. By using cross-tabs, a shift between pre- and postmeasurement is made visible. In all statistical analyses, a significance level of 5% ($p = \leq 0.05$) is assumed.

The nature of the data meets the conditions for the assumption of normality and asserts that the distribution of sample means

TABLE I
INFLUENCE STATIC/DYNAMIC ENVIRONMENT ON SOLVING PROGRAMMING TASKS CORRECTLY ($N = 29$)

		Task solved correctly									
Group	**n**	**Total**	**0**	**1**	**2**	**3**	**4**	**5**	**M**	**SD**	**Range**	**Mdn**
Static	17	30	2	4	7	4	0	0	0.35	0.19	0.00 – 0.60	0.40
Dynamic	12	46	0	0	2	4	0	6	0.77	0.25	0.40 – 1.00	0.80

Note. Static = Group that programmed in static environment; Dynamic = Group that programmed in dynamic environment; n = number of respondents; Total, Cumulative; M = average; SD = standard deviation; Range = spread in measurement; Mdn = median.

TABLE II
INFLUENCE STATIC/DYNAMIC ENVIRONMENT ON SRA APPLIED ($N = 29$)

		SRA applied									
Group	**n**	**Total**	**0**	**1**	**2**	**3**	**4**	**5**	**M**	**SD**	**Range**	**Mdn**
Static	17	35	6	4	0	2	0	5	0.85	0.84	0.00 – 2.00	0.60
Dynamic	12	44	0	0	2	3	4	3	1.36	0.49	0.80 – 2.00	1.20

Note. Static = Group that programmed in static environment; Dynamic = Group that programmed in dynamic environment; n = number of respondents; Total, Cumulative; M = average; SD = standard deviation; Range = spread in measurement; Mdn = median.

(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 or higher for a "large" effect size) [59].

A. Programming Task in Static or Dynamic Programming Environment Outflowing in Working Solutions

A comparison of the correctly solved programming problems shows (see Table I) that the group that programmed in the static environment solved 30 programming tasks correctly ($M = .35$) and the group that programmed in the dynamic environment solved 46 programming tasks correctly ($M = .77$).

Further examination of the data using the option "cross-tabs" showed that, comparing the two groups, a shift occurred with regard to the correct number of solved programming tasks. Table I shows the data for correctly solving the programming tasks divided over the two conditions.

It can be stated that, by comparing the static group with the dynamic group, more programming problems are solved by the dynamic group (increase = +16 more correctly solved) due to the robotics intervention and a different kind of task design.

From this data, it can be claimed that a dynamic programming environment indeed elicits that more programming problems are solved correctly. It is striking that within the dynamic group more complex programming problems are correctly solved. It is also remarkable that it becomes visible that there are several respondents from the dynamic group who have correctly solved all five programming problems.

B. Programming Tasks SRA Applied Within Static or Dynamic Problem Environment

An analysis of the data shows (see Table II) that the group that programmed within the static programming environment applied SRA 35 times ($M = 0.60$) and the group that programmed within a dynamic programming environment applied SRA 44 times ($M = 0.80$) when solving the programming tasks.

By means of "cross tabs" analysis, a shift is visible in the decision whether or not to apply SRA between the group that programmed in the static environment and the group that worked in the dynamic environment. By comparing the application of SRA of the static group with the dynamic group due to the offered environment, it can be stated that the dynamic group applied SRA more to solve programming problems (increase = +13,5% more SRA applied).

From this data, it can be claimed that a dynamic programming environment indeed elicits more use of SRA to solve programming problems. From the observations made during the execution, it is noticeable that the group that programmed in the dynamic environment applied SRA in a more direct approach to solving the programming problem in a comparison with the group that programmed in the static environment.

C. Analysis of the Quality of the SRA Application (Comparing Static and Dynamic Environment)

A qualitative analysis of the Lego EV-3 programs constructed by pupils indicates (see Table III) that there is a substantial difference in the application of SRA when comparing the group that programmed in the static environment with the group that programmed in the dynamic environment.

The group that programmed in the static environment, despite having had an introduction to the application possibilities of the SRA-approach, tried to solve the programming problem predominantly using a linear, sequential approach. However, it is clearly noticeable that this static environment does not invite to use SRA-programming. Pupils program toward the solution without the use of sensors, iterations, conditionals, and functions. The constructed Lego EV-3 programs showed that SRA is applied incidentally, but where SRA is applied, this is often not applied efficiently. The constructed SRA programs seem illogical, unnecessarily complex, and show a high level of unfocussedness in relation to the more functional and efficient use of the SRA-approach.

The group that programmed in the dynamic environment proceeds directly, and without hesitation, to the application of SRA. This is based on the most efficient way of using SRA, where no superfluous SRA-programming commands are included in the constructed program. Nowhere are unnecessary programming commands visible and the use of SRA has been effective and handled as effectively as possible.

TABLE III
INFLUENCE STATIC/DYNAMIC ENVIRONMENT ON QUALITY SRA APPLICATION ($N = 29$)

		Quality SRA			
Group	**n**	**M**	**SD**	**Range**	**Mdn**
Static	17	0.99	0.84	0.00 – 2.00	1.20
Dynamic	12	1.53	0.51	0.80 – 2.00	1.60

Note. Static = Group that programmed in static environment; Dynamic = Group that programmed in dynamic environment; n = number of respondents; Total, Cumulative; M = average; SD = standard deviation; Range = spread in measurement; Mdn = median.

TABLE IV
T-TEST ANALYSIS COMPARING STATIC/DYNAMIC ENVIRONMENT BASED ON SRA-USAGE

Variable	**t**	**df**	**p**	**CI**	**d**
Task solved correctly	-4.76	19.67	0.000	-0.60 – -0.23	1.86
SRA Applied	-2.10	26.26	0.023	-1.03 – -0.01	0.74
Quality SRA	-2.01	27.00	0.028	-1.10 – 0.01	1.20

Note. t = t-value; df = degrees of freedom; p = p-value; CI = confidence interval; d = effect size.

It can be claimed that the nature of the environment and the task design has an impact on whether or not SRA is applied and whether this has been done efficiently. This is visible on the basis of a qualitative analysis of the constructed Lego EV-3 programs when comparing the programming solutions of both groups. It is striking that the group that programmed in a dynamic environment applied SRA directly and constructed a more functional and effective program than the group that programmed in a static environment that frequently relied on linear programming. From observation and inventory, it becomes clear that the group that programmed in the dynamic environment used SRA much sooner and more directly than the group that programmed in the static environment.

D. Solving Programming Tasks

A comparison of the means makes visible (see Table I) that the group that programmed in the dynamic programming environment solved more programming problems ($M = 0.77$, $SD = 0.25$) than the group who solved programming problems in a static programming environment ($M = 0.35$, $SD = 0.19$). Analysis of the t-test (see Table IV) shows that there is a significant difference $t\ (20) =$ -4.76, $p = < 0.01$, $d = 1.86$. Since the value "0" is not within the confidence interval, 95% CI [-0.60, -0.23], this strengthens the assumption that programming in a dynamic environment ensures, by using SRA, that more programming problems are solved.

E. SRA Applied

A comparison of the means makes visible (see Table II) that the group that programmed within a dynamic programming environment makes more use of the application of SRA ($M =$ 1.36, $SD = 0.49$) than the group that programmed in a static programming environment ($M = 0.85$, $SD = 0.84$). Analysis of the t-test (see Table IV) shows that there is a significant difference $t\ (26) =$ -2.10, $p = < 0.02$, $d = 0.74$. Since the value 0 is within the confidence interval, 95% CI [-1.03, -0.01], this reinforces the assumption that programming in a dynamic environment increases significantly the application of SRA.

F. Quality of SRA

A comparison of the means makes visible (see Table III) that the group that programmed in the dynamic programming environment shows a higher quality level of SRA ($M = 1.53$, $SD =$ 0.51) than the group that programmed in a static programming environment ($M = 0.99$, $SD = 84$). Analysis of the t-test (see Table IV) shows that there is a significant difference $t\ (27) =$ -2.01, $p = < 0.03$, $d = 1.20$. But since the value "0" is within the confidence interval, 95% CI [-1.10, 0.01], we must conclude that programming in a dynamic programming environment does not lead to a higher quality level of SRA because the two environments are not comparable on this characteristic.

VI. CONCLUSION

From an interpretation of the available data, we can conclude that the influence of the problem environment and the type of task design scaffolds have an impact on evoking SRA-thinking. Pupils who solve programming assignments in a static task design do not, or hardly, use SRA-programming. To the opposite, it is remarkable that a dynamic task design immediately triggers the use of SRA-programming. Moreover, to successfully solve dynamic programming tasks, the application of SRA-programming is a necessity. In essence, SRA-thinking requires using characteristics of CT (e.g., algorithms, analysis, parallel thinking, pattern recognition, problem decomposition, routines, etc.). Where SRA-programming in both environments has been used, an analysis of the quality of SRA provides us with findings that the usage of SRA in a dynamic programming environment is of a higher quality level. From this, we conclude that solving dynamic problem tasks has an influence on strengthening SRA-thinking.

The hypothesis that pupils who solve programming assignments in a dynamic task design are able to activate SRA-thinking outflowing in working solutions can be substantiated and confirmed on the basis of the significant measurement results.

The hypothesis that pupils who solve programming assignments in a dynamic task design apply SRA-thinking more than pupils who solve programming assignments in a static task design can be substantiated and confirmed on the basis of the significant measurement results.

The hypothesis that pupils who solve programming tasks in a dynamic task design show problem solving of a qualitatively higher level than pupils who solve programming assignments in a static task design cannot be confirmed despite the significant measurement results. This is due to the fact that pupils in the static programming environment did not or hardly apply SRA, to which a comparison of this characteristic is not valid.

VII. Discussion

This article claims to find answers to whether the problem environment and task design is of significant influence to evoke SRA-programming among primary school pupils when programming robots.

Based on indications in the data, it can be assumed that programming using a dynamic task design evokes SRA-thinking remarkably and more explicitly in a comparison with programming in a static task design. This imposes specific demands on a programming environment that incorporates the added value of SRA-programming to enable PBL. Applying programmable robots as an educational tool stimulates higher order thinking and problem-solving skills, fostering pupils' conceptualization in meaningful authentic ways [36]. This claim is consistent with assumptions made by Slangen [10] who states that learning how to program robots is most beneficial when pupils can solve programming problems in an inquisitive, problem-solving environment. Scientifically, it is, however, the question which features make a programming environment useful for fostering pupils' conceptualization.

Our research shows that, in order to successfully and creatively solve complex robotics issues, it is clearly necessary for pupils to adopt a problem-solving approach in which they renew their previous programming experiences in solving challenging programming tasks. In order to do this, pupils have to develop problem-solving skills during programming, being able to devise a problem approach, choose a solution strategy, and apply the correct programming action. This is consistent with the allegations of Castledine and Chalmers [36] who state that when pupils are engaged in programming robots, making changes to repeated programming processes with the aim of solving a specific problem through strategically and reflective programming, they increase their problem-solving abilities and metacognitive skills. However, to develop a structured way to cope with challenging programming problems, the use of CT skills is inevitably [60]. CT sometimes refers to superimposed, metacognitive skills and sometimes they are operationalized as concrete programming concepts. The scientific question that arises is do we need a more focused definition of CT.

As we concluded before, robotic program problem environments that are subject to dynamic change offer a more cognitive challenging task environment than a static one. To evoke SRA-thinking, pupils need to call upon their disposition for ingenuity, originality, and insightfulness. It seems, therefore, important to further investigate how applying past knowledge to new situations, striving for accuracy, thinking flexibly, thinking about thinking, questioning, and posing problems [15] are of importance to solve a programming task, to learn to think in steps, and to use conditional reasoning to understand SRA-thinking better. With this in mind, a variation in the problem environment in which a robot has to fulfill its programming tasks can be image-forming and directional to the user, enabling a transfer from the mental disposition to the program to be created. This is in line with assertions made by Johansson and Balkenius [55] and Kitano *et al.* [57] who state that solving dynamic programming problems in an robotic environment in which an reactive approach and anticipatory programming strategies are more efficient.

VIII. Limitations and Follow-Up Research

There are some limitations and considerations as to why our findings indicate a significance level. The limited number of respondents in this article does not yet make it possible to generalize from the results obtained. It is, therefore, advisable to repeat this research with a larger number of participants.

In this article, we used Lego EV-3 robots with Mindstorms software. It would be worthwhile to investigate whether a different kind of programming environment shows similar results. It would also be valuable to investigate whether a programming environment with only a visual output can generate the same output as the currently used Lego programming environment, which is characterized by a tangible perceptible output.

It would also be worthwhile to further investigate to what extent the interventions of teachers have an influence, and to what extent the quality and intensity of these interventions lead to a further deepening of SRA-programming. Seen from the same perspective, it would be valuable to measure the impact of these teacher interventions on the development of CT skills among primary school pupils.

Acknowledgment

The authors would like to thank Bert van de Werfhorst, Heutink Netherlands for the donation of the required sets Lego Mindstorms.

References

[1] N. Yelland, "The future is now: A review of the literature on the use of computers in early childhood education (1994-2004)," *AACE J.*, vol. 13, no. 3, pp. 201–232, 2005.

[2] S. Edwards, "Identifying the factors that influence computer use in the early childhood classroom," *Australas. J. Educ. Technol.*, vol. 21, no. 2, pp. 192–210, 2005, doi:10.14742/ajet.1334.

[3] Y. B. Kafai and Q. Burke, "Computer programming goes back to school," *Phi Delta Kappan*, vol. 95, no. 1, pp. 61–65, 2013, doi:10.1177/003172171309500111.

[4] L. P. E. Toh, A. Causo, P.-W. Tzuo, I.-M. Chen, and S. H. Yeo, "A review on the use of robots in education and young children," *J. Educ. Technol. Soc.*, vol. 19, no. 2, pp. 148–163, 2016.

[5] Kennisnet, *Computational Thinking in Het Nederlandse onderwijs*. Zoetermeer, Netherlands: Kennisnet, 2016.

[6] J. Voogt, S. Brand-Gruwel, and J. Van Strien, "Effects of programming education on computational thinking: A review study," *Educ. Sci.*, vol. 2017, no. 1, pp. 1–33, 2017.

[7] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, 2006, doi:10.1145/1118178.1118215.

[8] D. Catlin and J. Woollard, "Educational robots and computational thinking," in *Proc. 4th Int. Conf. Robot. Educ.*, 2014, pp. 144–151.

[9] T. Sapounidis, S. Demetriadis, and I. Stamelos, "Evaluating children performance with graphical and tangible robot programming tools," *Pers. Ubiquitous Comput.*, vol. 19, no. 1, pp. 225–237, 2015, doi:10.1007/s00779-014-0774-3.

[10] L. Slangen, *Teaching Robotics in Primary School*. Eindhoven, Netherlands: Eindhoven Univ. Technol., 2016.

[11] L. Slangen, H. V. Keulen, and K. Gravemeijer, "What pupils can learn from working with robotic direct manipulation environments," *Int. J. Technol. Des. Educ.*, vol. 21, no. 4, pp. 449–469, 2011, doi:10.1007/s10798-010-9130-8.

[12] D. H. Jonassen, *Modeling With Technology: Mindtools for Conceptual Change*. Upper Saddle River, NJ, USA: Pearson Merrill Prentice Hall, 2006.

[13] J. Rekimoto, "Multiple-computer user interfaces: Beyond the desktop direct manipulation environments," in *Proc. Extended Abstr. Hum. Factors Comput. Syst.*, 2000, pp. 6–7, doi:10.1145/633292.633297.

[14] N. Fanchamps, L. Slangen, P. Hennissen, and M. Specht, "The influence of SRA programming on algorithmic thinking and self-efficacy using lego robotics in two types of instruction," *Int. J. Technol. Des. Educ.*, vol. 31, pp. 203–222, 2021, doi:10.1007/s10798-019-09559-9.

[15] A. Costa and B. Kallick, *Habits of Mind: Activating and Engaging*. Alexandria, VA, USA: ASCD, 2000.

[16] D. Barr, J. Harrison, and L. Conery, "Computational thinking: A digital age skill for everyone," *Learn. Lead. Technol.*, vol. 38, no. 6, pp. 20–23, 2011.

[17] A. Thijs, P. Fisser, and M. V. D. Hoeven, *Digitale Geletterdheid En 21e Eeuwse Vaardigheden in Het Funderend Onderwijs: Een Conceptueel Kader*. Enschede, Netherlands: SLO, 2014.

[18] E. Silk, C. Schunn, and R. Shoop, "Synchronized robot dancing: Motivating efficiency & meaning in problem-solving with robotics," *Robot. Mag. Carnegie Mellon Robot. Acad.*, vol. 17, pp. 74–77, 2009.

[19] SLO. *Curriculum Van De Toekomst*. Enschede, Netherlands: SLO, Jun. 2019. [Online]. Available: http://curriculumvandetoekomst.slo.nl/21e-eeuwse-vaardigheden

[20] I. Lee *et al.*, "Computational thinking for youth in practice," *ACM Inroads*, vol. 2, no. 1, pp. 32–37, 2011, doi:10.1145/1929887.1929902.

[21] M. U. Bers, L. Flannery, E. R. Kazakoff, and A. Sullivan, "Computational thinking and tinkering: Exploration of an early childhood robotics curriculum," *Comput. Educ.*, vol. 72, pp. 145–157, 2014, doi:10.1016/j.compedu.2013.10.020.

[22] A. Yadav, S. Gretter, J. Good, and T. McLean, "Computational thinking in teacher education," in *Emerging Research, Practice, and Policy on Computational Thinking*, P. Rich and C. Hodges, Eds. Berlin, Germany: Springer-Verlag, 2017, pp. 205–220, doi:10.1007/978-3-319-52691-1_13.

[23] L. Perković, A. Settle, S. Hwang, and J. Jones, "A framework for computational thinking across the curriculum," in *Proc. 15th Annu. Conf. Innov. Technol. Comp. Sci. Educ.*, 2010, pp. 123–127, doi:10.1145/1822090.1822126.

[24] A. Costa, *Developing Minds: A resource Book For Teaching Thinking*, 3rd ed. Alexandria, VA, USA: ERIC, 2001, pp. 1–592.

[25] J. D. Bransford, A. L. Brown, and R. R. Cocking, *How People Learn*. Washington, DC, USA: National Academy Press, 2000.

[26] H. L. Swanson, "Influence of metacognitive knowledge and aptitude on problem solving," *J. Educ. Psychol.*, vol. 82, no. 2, pp. 306–314, 1990, doi:10.1037/0022-0663.82.2.306.

[27] Hmelo-Silver, "Problem-based learning: What and how do students learn?," *Educ. Psychol. Rev.*, vol. 16, no. 3, pp. 235–266, 2004, doi:10.1023/B:EDPR.0000034022.16470.f3.

[28] K. Appleton, "Problem solving in science lessons: How students explore the problem space," *Res. Sci. Educ.*, vol. 25, no. 4, pp. 383–393, 1995, doi:10.1007/BF02357384.

[29] R. Moreno, G. Ozogul, and M. Reisslein, "Teaching with concrete and abstract visual representations: Effects on students' problem solving, problem representations, and learning perceptions," *J. Educ. Psychol.*, vol. 103, no. 1, 2011, Art. no. 32, doi:10.1037/a0021995.

[30] E. De Graaf and A. Kolmos, "Characteristics of problem-based learning," *Int. J. Eng. Educ.*, vol. 19, no. 5, pp. 657–662, 2003.

[31] W. Hung, D. H. Jonassen, and R. Liu, "Problem-based learning," *Handbook Res. Educ. Commun. Technol.*, vol. 3, no. 1, pp. 485–506, 2008.

[32] A. Yadav, H. Hong, and C. Stephenson, "Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms," *TechTrends*, vol. 60, pp. 565–568, 2016, doi:10.1007/s11528-016-0087-7.

[33] L. Slangen, *Techniek: Leren Door Doen! Didactiek en Bronnen Voor De Pabo*. Baarn, Netherlands: HB Uitgevers, 2009.

[34] J. R. Savery, *Overview of Problem-Based Learning: Definition and Distinctions, the Interdisciplinary*. West Lafayette, IN, USA: Purdue, 2015, pp. 5–15.

[35] D. H. Dolmans, W. De Grave, I. H. Wolfhagen, and C. P. Van Der Vleuten, "Problem-based learning: Future challenges for educational practice and research," *Med. Educ.*, vol. 39, no. 7, pp. 732–741, 2005, doi:10.1111/j.1365-2929.2005.02205.x.

[36] A.-R. Castledine and C. Chalmers, "LEGO robotics: An authentic problem solving tool?," *Des. Technol. Educ., An Int. J.*, vol. 16, no. 3, pp. 19–27, 2011.

[37] J. Lindh and T. Holgersson, "Does lego training stimulate pupils' ability to solve logical problems?," *Comput. Educ.*, vol. 49, no. 4, pp. 1097–1111, 2007, doi:10.1016/j.compedu.2005.12.008.

[38] B. Ricca, E. Lulis, and D. Bade, "Lego mindstorms and the growth of critical thinking," *Citeseer*, vol. 2006, no. 1, pp. 1–6, 2006.

[39] R. D. Pea and D. M. Kurland, "On the cognitive effects of learning computer programming," *New Ideas Psychol.,* vol. 2, no. 2, pp. 137–168, 1984, doi:10.1016/0732-118X(84)90018-7.

[40] P. V. Lith, *Masterclass Robotica*. Limbricht, Netherlands: Elektuur, 2006.

[41] C. Gregg, L. Tychonievich, J. Cohoon, and K. Hazelwood, "EcoSim: A language and experience teaching parallel programming in elementary school," in *Proc. 43rd ACM Tech. Symp. Comput. Sci. Educ.*, 2012, pp. 51–56, doi:10.1145/2157136.2157155.

[42] P. Wyeth, M. Venz, and G. Wyeth, "Scaffolding children's robot building and programming activities," *Lecture Notes Comput. Sci.*, vol. 3020, pp. 308–319, 2004, doi:10.1007/978-3-540-25940-4_27.

[43] L. Slangen, N. Fanchamps, and P. Kommers, "A case study about supporting the development of thinking by means of ICT and concretisation tools," *Int. J. Cont. Eng. Educ. Life-Long Learn.*, vol. 18, no. 3, pp. 305–322, 2008, doi:10.1504/IJCEELL.2008.018834.

[44] G. A. Demetriou, *Mobile Robotics in Education and Research*. Rijeka, Croatia: IntechOpen, 2011, pp. 27–48.

[45] M. Dragone *et al.*, "Robot soccer anywhere: Achieving persistent autonomous navigation, mapping, and object vision tracking in dynamic environments," in *Proc. OPTO-Ireland*, 2005, pp. 255–265, doi:10.1117/12.608404.

[46] B. Caci and A. D'Amico, "Children's cognitive abilities in construction and programming robots," in *Proc. 11th IEEE Int. Workshop Robot Hum. Interactive Commun.*, 2002, pp. 189–191, doi:10.1109/ROMAN.2002.1045620.

[47] L. Slangen and E. Rohaan, "Programmeren en robotica," in *Onderzoekend En Ontwerpend De Wereld Ontdekken*, T. Malmberg, E. Rohaan, S. Van Duijn, and R. Klapwijk, Eds. Groningen, Netherlands: Noordhoff Uitgevers bv, 2018, Art. no. 18.

[48] J. Lavonen, V. Meisalo, and M. Lattu, "Collaborative problem solving in a control technology learning environment, a pilot study," *Int. J. Technol. Des. Educ.*, vol. 12, no. 2, pp. 139–160, 2002, doi:10.1023/A:1015261004362.

[49] S. C. Kong, "A curriculum framework for implementing information technology in school education to foster information literacy," *Comput. Educ.*, vol. 51, no. 1, pp. 129–141, 2008, doi:10.1016/j.compedu.2007.04.005.

[50] P. Mosley and R. Kline, "Engaging students: A framework using LEGO robotics to teach problem solving," *Inf. Technol., Learn. Perform. J.*, vol. 24, no. 1, pp. 39–45, 2006.

[51] F. B. V. Benitti, "Exploring the educational potential of robotics in schools: A systematic review," *Comput. Educ.*, vol. 58, no. 3, pp. 978–988, 2012.

[52] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawai, and H. Matsubara, "Robocup: A challenge problem for AI and robotics," *in Proc. Robot Soccer World Cup*, 2005, vol. 1395, pp. 1–19, doi:10.1007/3-540-64473-3_46.

[53] O. Miglino, H. H. Lund, and M. Cardaci, "Robotics as an educational tool," *J. Interact. Learn. Res.*, vol. 10, no. 1, pp. 25–47, 1999.

[54] M. Asada, H. Kitano, I. Noda, and M. Veloso, "RoboCup: Today and tomorrow—What we have learned," *Artif. Intell.*, vol. 110, no. 2, pp. 193–214, 1999.

[55] B. Johansson and C. Balkenius, "An experimental study of anticipation in simple robot navigation," *Lecture Notes Comput. Sci.*, vol. 4520, pp. 365–378, 2007, doi:10.1007/978-3-540-74262-3_20.

[56] D.-H. Kim and J.-H. Kim, "A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer," *Robot. Auton. Syst.*, vol. 42, no. 1, pp. 17–30, 2003, doi:10.1016/S0921-8890(02)00311-1.

[57] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "Robocup: The robot world cup initiative," in *Proc. 1st Int. Conf. Auton. Agents*, 1995. pp. 340–347, doi:10.1145/267658.267738.

[58] D. Weintrop and U. Wilensky, "To block or not to block, that is the question: Students' perceptions of blocks-based programming," in *Proc. 14th Int. Conf. Interact. Des. Child.*, 2015, pp. 199–208, doi:10.1145/2771839.2771860.

[59] A. Field, *Discovering Statistics Using IBM SPSS Statistics*. Newbury Park, CA, USA: Sage, 2013.

[60] M. G. Voskoglou and S. Buckley, "Problem solving and computational thinking in a learning environment," *Egyptian Comput. Sci. J.*, vol. 36, no. 4, pp. 28–46, 2012.