Accepted Manuscript

Learning to code or coding to learn? A systematic review

Shahira Popat, Louise Starkey

PII: S0360-1315(18)30276-8

DOI: 10.1016/j.compedu.2018.10.005

Reference: CAE 3474

To appear in: *Computers & Education*

Received Date: 6 November 2017

Revised Date: 10 September 2018

Accepted Date: 5 October 2018

Please cite this article as: Popat S. & Starkey L., Learning to code or coding to learn? A systematic review, *Computers & Education* (2018), doi: https://doi.org/10.1016/j.compedu.2018.10.005.

Learning to Code or Coding to Learn? A Systematic Review

Shahira Popat and Louise Starkey*

Victoria University of Wellington, PO Box 600, Wellington, New Zealand.

*corresponding author. Email: louise.starkey@vuw.ac.nz

Abstract

The resurgence of computer programming in the school curriculum brings a promise of preparing students for the future that goes beyond just learning how to code. This study reviewed research to analyse educational outcomes for children learning to code at school. A systematic review was applied to identify relevant articles and a thematic analysis to synthesise the findings. Ten articles were included in the synthesis and an overarching model was developed which depicts the themes. The results demonstrate that although students are learning to code, a range of other educational outcomes can be learnt or practiced through the teaching of coding. These included mathematical problem-solving, critical thinking, social skills, self-management and academic skills. The review also identified the importance of instructional design for developing these educational outcomes through coding.

Highlights:

- Learning to code in schools enables students to learn skills beyond the coding itself
- Higher and lower order thinking skills can be developed when learning coding
- Curriculum and pedagogy influences the range of skills learnt when learning to code

Keywords**:** *coding, programming, school, computer, outcome, skills.*

Funding: This research did not receive any specific grant from funding agencies in the public, commercial or not-for-profit sectors.

Learning to Code or Coding to Learn? A Systematic Review

Abstract

The resurgence of computer programming in the school curriculum brings a promise of preparing students for the future that goes beyond just learning how to code. This study reviewed research to analyse educational outcomes for children learning to code at school. A systematic review was applied to identify relevant articles and a thematic analysis to synthesise the findings. Ten articles were included in the synthesis and an overarching model was developed which depicts the themes. The results demonstrate that although students are learning to code, a range of other educational outcomes can be learnt or practiced through the teaching of coding. These included mathematical problem-solving, critical thinking, social skills, self-management and academic skills. The review also identified the importance of instructional design for developing these educational outcomes through coding.

Keywords: coding, programming, school, computer, outcome, skills.

1. Introduction

Coding or computer programming was first developed with school children in mind in the 1960s with a vision that it would transform learning (Feurzeig, Papert, & Lawler, 2011). It was introduced into some primary schools through the programming software, LOGO during the 1980s then all but disappeared within a decade (Albion, 2016). The demise of computer programming was attributed to the focus of teaching moving to other computer skills, such as word processing and Internet searching which were regarded as more valuable at the time (Moreno-León, Robles, & Román-González, 2016; Pinkston, 2015). Since 2010 computing has been introduced in the primary school curriculum in Estonia, Greece, England and Australia (Albion, 2016, Balanskat & Englehart, 2014), enabled by the availability of

programming tools appropriate for younger learners (Resnick et al., 2009) and the development of computational thinking and coding curriculums (Kafai & Burke, 2013).

The revival in computer programming is evident in policy documents. For example, the K-12 framework for computer science education (National Research Council, 2012) and conceptual guidelines for computer science education (Alano et al. 2017) in the context of the USA with similar documents in other countries. The global revival is attributed to three reasons. The first is an economic drive of preparing the future workforce with knowledge of computer programming both to supply the IT industry and more generally for workers of the future (Balanskat & Englehart, 2014). The second is an entrepreneurial ideal of students learning to be 'producers' of innovation (eg. Kalelioğlu, 2015). The third reason situates the knowledge and skills gained through learning computer programming as 21st century competencies that everyone should learn (Wing, 2006). The third reason provides the focus of this review, an exploration of the competencies gained through learning computer programming.

Lye and Koh (2014) suggest that programming can expose students to computational thinking. While the scope of computational thinking is debated, Wing (2006) defined it as a fundamental analytical skill that should be learnt alongside reading, writing and mathematics from an early age. She also said it goes beyond a narrow focus of learning programming syntax and develops capabilities to think conceptually and problem solve at multiple levels of abstraction.

A growing body of research has examined computer science or computational thinking based outcomes when children are learning to code (Grover & Pea, 2013, Lye and Koh, 2014,). However, little research has explored outcomes beyond computer science and computational thinking. Researchers have suggested that the inclusion of coding in the school curriculum

provides a range of learning outcomes which can be applied to areas other than computer science (Kalelioğlu, 2015; Moreno-León et al., 2016, Resnick et al., 2009, Wing, 2006). This brings into question whether students who are learning to code are just learning coding and computational thinking, or through the process of coding they are learning other skills. This review explores the empirical evidence to determine whether students who are being taught coding are learning more than computational thinking or computer programming in other words, are students learning to code or coding to learn?

This review focussed on the following research question: What educational outcomes, beyond computational thinking or programming skills can be identified as a result of learning how to code in school?

2. Research method

A systematic review process as outlined by Jesson, Matheson & Lacey (2011) was applied to gather, synthesise and appraise the findings of studies which explore the relationship between coding and non-coding specific educational outcomes (Table 1).

Table 1. Process of systematic review.

Aim	To determine if coding influences student outcomes
Search strategy	Booelian search using: Coding OR programming OR programing (Title), AND school OR children (Abstract), AND skills OR thinking (Abstract)
Inclusion criteria	Research focused on the link between learning coding/programming and other learning outcomes. Full text, peer-reviewed, scholarly articles, empirical research.
Exclusion criteria	Studies on: tertiary or university students, describing methods of teaching coding, programming related outcomes, learning technologies other than coding and literature reviews.
Quality appraisal	Adapted from Critical Appraisal Skills Programme (CASP) checklist and notes on validity and reliability from Johnson & Christensen, 2016.
Data extraction	Read studies and collect relevant information.
Synthesis of data	Create a table of evidence of educational outcomes related to coding from the data extraction which identifies themes, similarities and differences.

Report	Results analysed and summarised in a model created to demonstrate the impact of coding on educational outcomes.

The main criteria for inclusion was empirical research that explores skills or knowledge learnt within a school coding/programming curriculum that are not specific to learning how to code. Specific education databases were initially searched, these included A+ Education, Education Source (EBSCO), and ProQuest Central. These databases together provided access to an extensive and broad range of education journals and indexed conference proceedings. The initial search included the key words coding OR programming. This identified a large number of articles focusing on educational programmes rather than computer programming or how data was coded within educational research rather than the teaching of coding. Relevant articles retrieved in this initial search included either coding, programming or programing in the title. Therefore, a filter was used to limit the keywords to within the title. It could be possible that not all relevant research would have these words in the title. Other key terms which through a process of trialling terms to optimise results included requiring the terms school OR children AND skill OR thinking in the abstract. Filters were used to include only articles from peer-reviewed sources. As coding was first developed for students in the 1960s there was no restriction placed on the date of publishing. Most articles retrieved were published between 2012 and 2016.

The search strategy identified 172 potentially relevant research articles in October 2017. Due to the risk of potentially excluding relevant articles exclusions was made manually through three stages. The first involved reading the title of each article, those that appeared possibly relevant were scrutinised further through the reading of the abstract or preview then a third stage involved reading full articles. The articles that were excluded included those that described teaching programmes or practices, or evaluated students' achievement in computer programming. One potential article was excluded as the extended abstract was in English and

the full article was only available in Turkish. The search strategy was undertaken independently by two reviewers to reduce individual bias as recommended by Jesson et al. (2011). There were no disagreements to be found reflecting the transparency of the quality criteria. Based on this strategy, 10 articles were identified, and evidence evaluated on the overall validity of these. The search results are shown in Table 2 and an overview of each article is summarised in appendix 1. Each of the articles were then evaluated using an adapted version of the critical appraisal skills programme checklist (Kuper, Lingard, & Levinson, 2008) as adapted for mixed methods reviews by Tondeur et al. (2011) to assist in assessing quality of the research.

Table 2. Search results (October 2017)

Database	Total hits	Results after applying exclusion criteria
A+ Education	14	0
Education Source (EBSCO)	102	7
ProQuest Central (ERIC)	56	3
Total	172	10

Retrieved articles from ProQuest Central exclude those already retrieved from Education Source. Some full text research articles were not able to be retrieved online.

The next process of systematic review was to extract data; this was completed by reading the studies and any key information was recorded and summarised. To identify patterns across the articles with regards to the educational outcomes influenced by coding, it seemed appropriate to use a thematic approach when synthesising the data. Educational aspects identified in each article were recorded and grouped into themes. Both qualitative and quantitative data were analysed in the same way, through the identification of reported outcomes. The themes were then refined, for example, problem-solving" was merged with "mathematical concepts" because the problems were solved through the application of

mathematical concepts. The articles were re-read considering the themes that had been listed and relationships between the studies identified.

Bloom's revised taxonomy was used as a framework to compare the cognitive educational outcomes against the levels of thinking. The taxonomy has been widely used in the educational community to compare and categorise educational outcomes (Anderson & Krathwohl, 2001). The six levels within the cognitive domain are organised into two levels of thinking; Lower-order thinking skills of remembering, understanding and applying and higher-order thinking skills of analysing, evaluating and creating

The cognitive skills identified by this review were assessed against the six cognitive domains using a checklist of verbs from Anderson and Krathwohl (2001) as a guideline and categorised into lower-thinking skills and higher-thinking skills to assess the breadth and depth of the skills identified. For example, when thinking critically, if students tested and evaluated their code, this was categorised as *evaluating* and provided evidence of higher-order thinking skills.

The other key themes outside of the cognitive domain included, social skills/collaboration and self-management. The articles were re-read considering the themes that had been listed and relationships between the studies identified.

3. Results

Ten articles were used in the review and included quantitative data from school students, aged between 5 and 17 years old, and qualitative data from primary school students and teachers. Articles are from a range of countries: from the USA, New Zealand, Greece, Ireland, Turkey and Spain which could infer global interest and contemporary importance of the topic, the studies were published between 1988 and 2017. An overview with summary information regarding each of the selected articles is included in appendix 1.

3.1 Synthesis of findings

The educational outcome of learning to code was evident in all ten studies however, this research has explored outcomes beyond computer science and computational thinking. Based on the synthesis, five key themes were identified relating to other educational outcomes from the teaching of coding. A summary of the synthesis of data and themes identified are shown in Table 3. Across the literature there was evidence to suggest that the curriculum and pedagogical contexts of the research studies had an impact on the learning outcomes, therefore this is also discussed as a sixth theme.

Table 3. Educational outcomes identified from studies

Study	Educational outcomes excluding coding skills				
First author	Problem-solving through mathematical concepts	Social skills including collaboration	Self-management/ Active learning	Critical thinking	Academic skills (NOT including mathematical or computer science/ programming related skills)
Bernardo	x				
Falloon	x	x	x	x	
Fessakis	x	x		x	
Hayes	x			x	x
Kalelioğlu & Gülbahar	x		x		
Kalelioğlu	x			x	
Miller	x				
Palumbo	x			x	
Sáez-López,	x		x	x	x
Psycharis	x			x	

3.2 Key theme 1: Problem-solving through mathematical concepts.

This theme was evident within all ten studies. Nine studies actively tested for problem-solving ability which had to be applied through the mathematical concepts presented within the coding tasks set. The tenth study recognised that in the process of learning how to program, students learnt mathematical skills which enhanced problem-solving. In this review, problem-solving was defined as presenting a problem and students carrying out a solution to the problem. Coding or computational problem-solving skills were those where a student could design and run an algorithmic procedure (i.e. telling the computer what to do to solve the problem). However, as an educational outcome outside of programming, problem-solving through mathematical concepts was a more generic skill that could be analysable in different situations. For example, logical reasoning, which included the ability to interpret patterns, number sequences or the relationship between shapes to solve problems.

There is an overlap between mathematical concepts and coding, in the study of Fessakis, Gouli and Mavroudi (2013), five and six-year olds were asked to solve a series of problems using programming software. For example, navigating a ladybug through a maze in as few commands as possible applying concepts of orientation, angle rotation, counting and measuring. Eight out of ten students could complete all the activities. Falloon (2016) also identified young students problem-solving through mathematics within the programming of Scratch. Similarly, in Kalelioğlu (2015), activities which practice mathematical problem-solving using code.org were given to Year 4 students. All students could complete three out of the nine activities (however as tasks became more complex, less students could complete them). The early studies by Bernardo and Morris (1994) and Palumbo and Reed (1991) measured problem-solving ability and found that learning coding improved mathematical problem-solving significantly more than control groups who did not receive specific programming instruction. However, the evidence in other studies is less convincing.

The study by Miller, Kelly and Kelly (1988) found that the boys in the treatment group achieved higher in problem-solving than the boys in the control group, with no difference between the two groups of girls. This was carried out without pretesting; therefore, the differences may not be attributed to the learning of programming. In a quasi-experimental study, Psycharis and Kallia (2017) found that teaching Mathematics in conjunction with computer programming significantly enhanced mathematical problem-solving, although the improvement was not statistically significantly more than the control group that learnt mathematics without computer programming. Falloon (2016) and Kalelioğlu and Gülbahar (2014) found that there was no significant difference in problem-solving competence resulting from the use of Scratch in primary schools. Hayes and Stewart's (2016) study compared cognitive change in students using Scratch with those using a derived relational responding (DRR) training programme and concluded those using Scratch improved less than those in the DRR programme. While computer programming can improve mathematical problem-solving, the extent of learning may depend on how closely aligned the measurement is with the teaching programme.

As problem-solving was a cognitive skill, Bloom's revised taxonomy was used to determine the depth and breadth of this educational outcome. It was clear that students had to understand the problem and apply the concepts. However students *analysed* the problem, for example when *determining* how to use as few commands as possible therefore this demonstrated higher-order thinking skills. There was also an overlap between problem-solving and critical thinking which is discussed in key theme 4.

Problem-solving was tested through the ability to solve mathematical problems. Overall the evidence suggests that problem-solving through mathematical concepts can be an educational outcome of learning coding. However, the learning of mathematical problem solving may not be better attained through the learning of coding than through direct teaching. Therefore the

pedagogical intentions of the teacher and what is measured in the research are important considerations. For example, in the study by Kalelioğlu (2015), new ways of solving problems learnt through coding were identified by the students that were interviewed, although this was not identified in the measures of problem-solving.

3.3 Key theme 2: Social skills including collaboration.

Evidence in two studies directly refer to social skills including collaboration as a feature of learning computer programming or coding. Social skills refer to the ability to communicate effectively with other people. This can be through working positively with others. This was not necessarily a planned learning outcome but was reinforced within the classroom context. Kalelioğlu (2015) and Fessakis et al. (2013) identified that students would share their coding with others who needed help. One student stated “we explained what we done to those who couldn’t make progress. We helped those left behind when we moved forward” (Kalelioğlu, 2015, p. 206). Fessakis et al. (2013) also found that there was dialogue development among students and discussion around the activities and the issues related to the tasks. These observations advocate that coding allows for interaction among students and therefore, it is assumed, the development of social skills. Falloon (2016) suggests that to facilitate collaboration, students should be organised into pairs to share their knowledge and understanding and encourage acceptance of other people’s views. While collaboration appeared to be a feature of learning, there were no measurements to ascertain whether this was a skill that was developed because of learning computer programming or was a skill that was already developed within the context of the classrooms being studied.

3.4 Key theme 3: Self-management and active learning.

A theme derived from three out of ten studies was self-management and active learning. Only one out of these three studies measured active learning as an outcome, the other two found

evidence through observations and qualitative data. This theme is defined as the process where students can engage in the activities and have control over their own learning. Kalelioğlu (2015) says that students tried to complete more stages and more levels at their own pace using code.org in comparison to their normal classroom activities. The teacher observed that students were willing to continue through the stages of activities and those who were not previously interested in the lesson participated and showed “amazing” progress when using code.org. A student stated, “I want to complete all stages of code.org site” (Kalelioğlu, 2015, p. 208). Students were taught content through videos as well as how to review their progress. Falloon (2016) observed that the young students in his study spent a great deal of time discussing and planning what they had to do. Through observation and questionnaires, Sáez-López, Román-González and Vázquez-Cano (2016) concluded that children recognised their own active approach to learning. As the theme suggests coding can allow students to become active learners. However, there is also recognition in the studies that pedagogical aspects such as specific teaching and task design is required to develop student understanding of how to successfully carry out learning activities at their own pace to enable them to be self-managing and active learners.

3.5 Key theme 4: Critical thinking.

Critical thinking was identified in five of the studies. There is an overlap between critical thinking and problem-solving, as to solve problems children must think critically and apply reasoning. In this review, the theme of problem-solving is the act of solving the problem through mathematical concepts whereas critical thinking is defined as the ability to carry out some or all the following; analyse a task, create and implement a plan, evaluate results, identify and apply actions needed to improve or correct performance, and assess whether the desired outcome is achieved. Critical thinking was identified as a feature across different studies. Falloon (2016) identified that students using Scratch predicted a possible outcome of

running their code before they ran it. Kalelioğlu (2015) tested aspects of critical thinking quantitatively, a slight increase was found in the average difference between pre-and post-test results for evaluation. This was assessed as the ability to step back and think critically about how problems are solved. Similarly, Palumbo et al (1991) used a critical thinking appraisal and found increases in analysis and interpretation of information in comparison to the control group. In the study conducted by Sáez-López, et al. (2016) observations of students and results from the student questionnaire noted that students could analyse historical and artistic content in paintings with the use of resources from the Scratch application. In Fessakis et al. (2013), students were encouraged to find better alternative solutions than students who had completed the task before them hence improving performance. In addition, a student expressed learning to think critically stating "I made some mistakes in the process… but I've learned to use my logic when we use this program" (Kalelioğlu, 2015, p. 207), thus implying that they could correct their performance. Improved reasoning skills have been identified as an outcome of learning computer programming (Hayes & Stewart, 2016; Psycharis & Kallia, 2017).

The evidence suggests that aspects of critical thinking can be developed through coding activities and task design should encourage students to test, evaluate and modify code to develop their critical thinking skills. Being able to *test*, *evaluate* and *modify* code was assessed against Bloom's revised taxonomy as higher-order thinking skills.

3.6 Key theme 5: Academic skills or knowledge (not including mathematical or computer related skills).

Academic skills or knowledge was identified in two out of the ten studies and includes increasing the level of competence in school-related subject areas. Subject areas not included in this theme are mathematics and coding/computer science as these are evident in all studies

and have been discussed separately. In Hayes and Stewart (2016) students, aged 10 – 11, who received instruction using Scratch programming were tested on academic skills such as reading and spelling. Pre-test and post-test results showed an improvement within these academic skills however it is unclear of the significance of the improvements due to the methodology used. Sáez-López, et al. (2016) implemented programming into the art history curriculum and found that students could understand and comprehend a range of art and history concepts using programming software. Therefore, there is some implication that coding can influence academic skill however this was within the pedagogical design of the activity and when analysed against Bloom's revised taxonomy this seemed to be with regards to lower order thinking for example, recalling information.

3.7 Curriculum and pedagogical design

The curriculum and pedagogical design includes what is taught in coding and how it is taught. This theme relates to an aspect that influences educational outcomes rather than an outcome in itself. There was evidence across the first five themes to suggest that the curriculum and pedagogical design was important in influencing the learning outcomes identified. This included teaching coding in subject areas other than computing, task design or by sitting in pairs or groups. For example, integrating coding into subject areas such as mathematics and art history showed that students not only made significant gains in programming but also in course related content (Psycharis & Kallia, 2017; Sáez-López et al., 2016). Certain coding tasks using Scratch programming software were not only used to complement the teaching of mathematical concepts such as angles, direction and distance, but provided opportunities for mathematical problem-solving and critical thinking. This included logical reasoning to analyse and solve problems, for example, by making predictions to identify and correct errors before creating or testing code (Falloon, 2016). The research also identified that how students were organised in class allowed for the development of social

skills such as communication and collaboration when completing coding tasks. For example, students sitting beside each other rather than opposite one another allowed students to help one another if they were stuck (Falloon, 2016; Kalelioğlu, 2015; Fessakis, 2013). Some coding tasks provided opportunities for self-directed learning, for example, code.org provided video tutorials which students watched to teach themselves how to complete tasks and could work through these at their own pace monitoring their progress along the way (Kalelioğlu, 2015). It was also shown that teacher explanation, support and modelling certain behaviour was required to influence educational outcomes. For example, teachers who modelled peer support, by paraphrasing questions students could ask, developed communication and collaborative skills (Falloon, 2016). Fessakis et al. (2013) identified the teacher as key in facilitating learning, for example, if a student couldn't solve a problem they could ask for help, the teacher also encouraged students to complete coding tasks. Falloon, (2016) said teacher explanation was critical in developing social skills and Kalelioğlu, (2015) stressed the importance of teachers explaining more complex activities to help students. It can be explicitly seen from the literature how the curriculum and pedagogical contexts influenced learning outcomes.

The findings were used to develop an overarching model specific to the influence of coding on educational outcomes. This was the last step in the systematic review process.

3.8 The overarching model

Learning to code is at the centre of the overarching model (Figure 1) as this is a key outcome when learning through programming. Sáez-López, et al. (2016) applied a quantitative test which found that there are significant improvements in students' ability to understand programming concepts and computational practices after using Scratch in the classroom. However, the studies provide evidence that other educational outcomes can be influenced by

the design of coding activities and the classroom setting. These can be nurtured through curriculum and pedagogical design within the classroom setting and through teacher facilitation. The themes of educational outcomes identified are grouped under three broad headings; lower order thinking skills such as understanding and comprehending, higher order thinking skills of critical thinking and problem-solving through mathematical concepts and personal skills which include social skills and self-management. Lower-order and higher-order thinking skills were used for the cognitive skills identified which were analysed against Bloom's revised taxonomy. Personal skills was seen as suitable heading for communicating, collaborating and self-management as these skills seemed to be used naturally through students own experiences and practice. The skills under each heading are interrelated. For example, to solve problems students must think critically and may do so collaboratively.

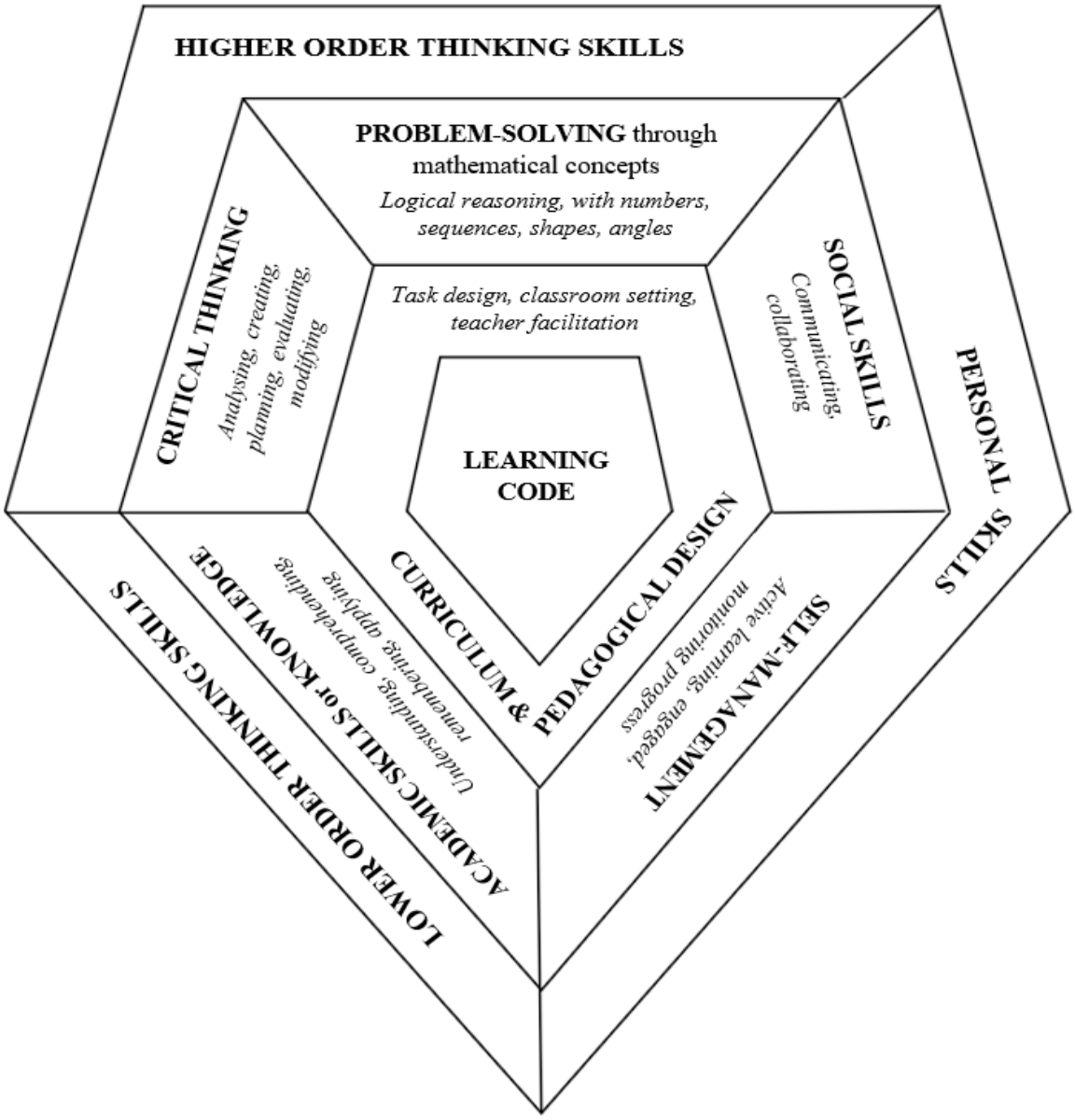

Figure 1. Model depicting the influence of coding on educational outcomes.

Overall, the literature suggests that through learning to code, students use a range of skills beyond coding. However, these skills are developed further through the deliberate teaching and inclusion in the curriculum and pedagogical design.

4. Discussion

This review focused on the influence of coding on educational outcomes for school children. The synthesis of findings resulted in an overarching model that was developed to depict an

overall visual representation of the key outcomes identified. This section discusses the results and overarching model in relation to other seminal pieces of work and the current state of programming in schools.

Papert (1980), had a vision that children would program the computer rather than the computer program the child. He felt that school children would learn by discovery and establish deep ideas from science, mathematics and technology. Mayer (1988) stated that studies during this time did not match the powerful claims made by those such as Papert, and research identified specific programming skills being gained but were less certain about the development of general skills. In this review learning to code is a key outcome in all ten studies. This is comparable to other studies (For example, Mayer & Fay, 1987, Lye & Koh, 2014), therefore, learning to code sits in the centre of the overarching model.

The aim of this review was to find evidence of educational outcomes beyond coding that were influenced by learning to code, these are depicted in the third level of the overarching model and aligned with three types of skills: higher order, lower order and personal skills.

4.1 Higher order skills

The first higher order skill identified was problem-solving through mathematical concepts. While problem solving is an essential element of computational thinking, the development of mathematical problem solving was a theme identified in this systematic review. Problem-solving as a mathematical process involves the use of mathematical skills such as numbers and measurement to solve problems. For example, developing problem-solving ability with regards to algebraic word problems (Soloway, Lochhead & Clement, 1982). It is identified as a higher order thinking skill in the overarching model, based on Bloom's revised taxonomy (Anderson & Krathwohl, 2001).

The mathematical problem-solving discovered in this review and previous research is not necessarily what is envisaged by current K-12 computer science related standards. The K-12 computer science framework said its vision is for students to "learn new approaches to problem solving that harness the power of computational thinking to become both users and creators of computing technology" (Alano et al., 2017, p.9). Computational thinking is broader than just mathematically well-defined problems with analysable solutions (Wing, 2006). This review did not investigate computational thinking but it found that mathematical problem-solving was a key educational outcome through learning code which may be due to the close alignment with computational problem solving.

The reviewed literature suggests that mathematical problem solving is an educational outcome when learning to code. However, while mathematical skills, such as the use of geometry to solve problems, may be enhanced by learning coding, other programmes of learning yield better or the same improvement (Hayes & Stewart, 2016; Kalelioğlu & Gülbahar, 2014). Therefore, if the academic aim is for students to learn mathematical problem solving, teaching these skills directly is more effective than learning these through coding.

Mayer's depiction of the chain of cognitive changes include learning to think in domains outside of programming (1988). Research that has explored the application of problem-solving skills learnt while learning to code has found skill transfer is most likely to occur when the tasks or contexts are similar to those within which the learning occurred (Mayer, 1988, Miller et al., 1988, Bernardo et al., 2015). Therefore, higher order thinking skills learnt through coding could be used in other situations when the context was similar.

The pedagogical processes while learning mathematical problem solving through coding is important. The mathematical problem-solving skills were developed through set tasks with

support from teachers and collaboration with other students which links this theme to other themes. In addition, while learning coding deliberate teaching is required to identify and correct misconceptions. Students who have misconceptions have been found to not improve their problem-solving ability compared to peers without misconceptions (Mayer and Fay,1987). This suggests that there is a need for teachers to intervene when misconceptions occur and to consider opportunities for collaborative learning when developing mathematical problem solving.

The second higher order skill identified by this review was critical thinking. Critical thinking was grouped under higher order thinking skills in the overarching model based on Bloom's revised taxonomy and included skills such as create and evaluate. While thinking creatively can be a key skill developed through coding (Resnick et al., 2015) this was not identified by the studies reviewed as many "learn to code" initiatives do not support the use of creative expression (Resnick et al., 2015). This is evident in this review, where most studies asked students to program movements of a character navigating through obstacles toward a goal, requiring evaluation skills but constrains creative thinking. Exposing students to computer science principles through programming in other subject areas such as music can allow for better articulation of broader uses of problem-solving and creative thinking skills (Alano et al., 2017). Therefore, although there is evidence of critical thinking, this needs to be encouraged further through the curriculum and pedagogical design (Falloon et al. 2016; Resnick et al. 2015; Thompson et al. 2008).

4.2 Lower order thinking skills

Generic academic skills were defined as a lower order thinking skill as it was based on the ability to understand and apply content and the scope of this review excluded understanding and applying coding. Academic skills beyond coding would only be an outcome if coding

was integrated into other subject areas. While the integration itself may require higher order thinking skills, the studies in this review were limited to remembering, understanding, comprehending or applying content knowledge. National and international information on the implementation of standards in computer education guide teachers to structure their programs to integrate computer science content such as programming into other content areas (Alano et al., 2017; TKI, 2017) to prepare students to apply computer science in authentic contexts. There seem to be few studies in schools that have considered the influence of learning coding on subject areas outside of mathematics and computer science (Alano et al., 2017).

4.3 Personal skills

Personal skills developed through coding included social skills and self management or active learning. Resnick and Segal (2015) perceive coding as a new type of literacy and personal expression rather than a set of technical skills, which like learning to write is valuable to everyone. They see coding as a way for people to express and share their ideas and work collaboratively, therefore requiring social skills. However, there is limited consensus on the importance of social skills in the context of learning computer science (Webb et al. 2017).

Communication and collaboration are included in Australian and Polish computer science curriculum (Webb et al., 2017) and a core concept of the K-12 computer science framework (Alano et al., 2017). This review found social skills such as the ability to communicate and work with others was an incidental learning outcome of learning to code that was influenced by curriculum and pedagogical design with one study identifying the deliberate teaching of collaboration skills while children learning to code (Falloon et al., 2016).

The self-management and active learning theme aligns with Papert's views of learning programming, where students take control of their own learning and monitor their own progress. Fessakis et al. (2013), found that four out of ten students progressed through trial

and error. However, there were also times where they had to ask for help which not only links to their use of social skills but to the support that is needed that goes beyond pure discovery as advocated by Papert (1980). The K-12 computer science framework suggests that teachers should scaffold active learning experiences to support and guide students to deeper engagement and higher levels of thinking, particularly at Pre-K (Alano et al., 2017). Thus while learning coding can enable students to apply self-management and active learning skills, there is an important role for teachers to design learning experiences that enable students to develop these skills.

4.4 Curriculum and pedagogical design, the glue in the overarching model

Curriculum and pedagogical design was the most important feature across the literature, providing the bridge between learning to code and the development of key educational outcomes. This was regarded as significant across the themes and therefore included in the second level of the overarching model. This level of the model is significant as it links not only to learning to code with individual educational outcomes identified but also across educational outcomes. For example, the learning activity studied may have been designed to teach children not only coding, but also art history (Sáez-López, et al, 2016) or mathematical problem solving (Bernardo & Morris, 1994). Guidance, direct instruction and explanation are required to ensure effective learning in computer science, rather than discovery learning without deliberate or organised teaching (Hagge, 2017; Mayer, 1988), task design can support active learning (Xia, 2017) and teacher guided instruction and group work can enable students to apply mathematical concepts (Lambic, 2010), improve communication and interaction with others (Pellas & Peroutseas, 2017). The educational outcomes from learning to code are determined by the curriculum and pedagogical design.

The educational outcomes identified by this study are supported from previous research. However, prior research has generally looked at one specific outcome such as problem-solving, whereas this study has created an overarching model of outcomes identified within the studies located through systematic review. This review therefore provides an overview of the educational outcomes developed though learning coding. An important factor across these studies is the curriculum and pedagogical design of learning activities.

5. Limitations

The search for empirical evidence was limited to three significant databases and key word searches within the title and abstract published up to 2017 which may have excluded relevant research articles. Having two reviewers reduced bias that can be inherent with single person systematic reviews. With regards to quality one study was not regarded as having a clear rationale for the method of research used and the programming group was the control group. The study was included in the review as there were positive influences of programming on educational outcomes but the significance of these are unknown.

6. Recommendations for further research

Two areas require further exploration. The first is the second level of the overarching model (curriculum and pedagogical design) which suggests that specific teaching is required to learn coding and to positively influence other educational outcomes. This element was not the aim of this review so was not researched in detail. The reviewed articles studies did not distinguish between the intended learning outcomes of teaching programming and incidental generic skills that were inherent in the pedagogical approach of the teachers or context of the learning environment.

The second area for further research is the application of specific coding skills outside of coding and the limitations of short time frames. Studies suggest that using the skills developed outside of the programming domain are most likely to occur for skills similar to those learnt in programming. Some limitations identified in the studies reviewed, discussed the length of the programming instruction as important particularly when assessing the use of skills in areas other than computer science, this is also supported by seminal pieces of work (Pea & Kurland, 1984; Mayer, 1988; Miller et al., 1988). Pea and Kurland (1984) suggest that longitudinal studies are required, in high school programmes students may spend 30 – 50 hours on coding per year, in comparison to the 1500+ hours by experienced programmers. The pedagogical approach and content of learning tasks is an area that is worthy of consideration when designing future studies.

7. Conclusion

The review provides a summary of research that explores the influence of coding on educational outcomes which were classified as higher order thinking, lower order thinking and personal skills which were incorporated into an overarching model. The model provided specific examples of each of these broad educational outcomes under five themes; critical thinking, problem-solving, social skills, self-management and academic skills or knowledge. This, along with the recommendations for further research provides insight into how and why coding influences educational outcomes. Students learning to code can achieve broader educational outcomes, and therefore coding to learn through the thoughtful design of teaching programmes.

References

Alano, J., Babb, D., Bell, J., Booker-Dwyer, T., DeLyser, L.A., McMunn Dooley, C., Phillips, R. (2017). *The K-12 computer science framework.* Retrieved from https://k12cs.org/

Albion, P. (2016). *The second coming of coding: will it bring rapture or rejection?* In S. Prestidge, & P. Albion (Eds.), If, *Australian Council for Computers in Education 2016 conference* (pp. 1-8). Brisbane: ACCE. Retrieved from: http://conference.acce.edu.au/index.php/acce/acce2016/paper/view/49

Anderson, L. W., & Krathwohl, D. R. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives.* New York: Longman.

Balanskat, A. & Englehart, K., (2014). *Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe.* Brussels: European Schoolnet. Retrieved from: http://www.eun.org/c/document_library/get_file?uuid=521cb928-6ec4-4a86-b522-9d8fd5cf60ce&groupId=43887

Bernardo, M., & Morris, J. (1994). Transfer effects of a high school computer programming course on mathematical modeling, procedural comprehension, and verbal problem solution. *Journal of Research on Computing in Education, 26*(4), 523-536. http://dx.doi.org/10.1080/08886504.1994.10782108

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *Journal of Computer Assisted Learning, 32*(6), 576-593. http://dx.doi.org/10.1111/jcal.12155

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63,* 87-97. https://doi.org/10.1016/j.compedu.2012.11.016

Feurzeig, W., Papert, S. A., & Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments, 19*(5), 487-501. http://dx.doi.org/10.1080/10494820903520040

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher, 42*(1), 38-43. http://dx.doi:10.3102/0013189X12463051

Hagge, J. (2017). Scratching beyond the surface of literacy: Programming for early adolescent gifted students. *Gifted Child Today, 40*(3), 154-162. https://doi.org/10.1177/1076217517707233

Hayes, J., & Stewart, I. (2016). Comparing the effects of derived relational training and computer coding on intellectual potential in school-age children. *British Journal of Educational Psychology, 86*(3), 397-411. http://dx.doi.org/10.1111/bjep.12114

Jesson, J. K., Matheson, L., & Lacey, F. M. (2011). *Doing your literature review: Traditional and systematic techniques*. Thousand Oaks, California: Sage.

Johnson, R. B., & Christensen, L. (2017). *Educational research: Quantitative, qualitative and mixed approaches* (6th Ed.). Thousand Oaks, CA: Sage.

Kafai, Y. B., & Burke, Q. (2013). Computer Programming Goes Back to School. *Phi Delta Kappan, 95*(1), 61-65. https://doi.org/10.1177/003172171309500111

Kalelioğlu, F., & Gülbahar, Y. (2014). The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. *Informatics in Education, 13*(1), 33-50.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior, 52*, 200. https://doi.org/10.1016/j.chb.2015.05.047

Kuper, A., Lingard, L., & Levinson, W. (2008). Critically appraising qualitative research. *BMJ*, 337, a1035. https://doi.org/10.1136/bmj.a1035

Lambic, D. (2010). Presenting practical application of Mathematics by the use of programming software with easily available visual components. *Teaching Mathematics and Its Applications, 30*, 10-18. https://doi.org/10.1093/teamat/hrq014

Lye, S. & Koh, J. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*. *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Mayer, R.E., & Fay, A.L. (1987). A chain of cognitive changes with learning to program in LOGO. *Journal of Educational Psychology. 79(3),* 269-279. http://dx.doi.org/10.1037/0022-0663.79.3.269

Mayer, R. E. (1988). *Teaching and learning computer programming: Multiple research perspectives.* Mahwah, NJ: Erlbaum.

Miller, R. B., Kelly, G. N., & Kelly, J. T. (1988). Effects of Logo computer programming experience on problem solving and spatial relations ability. *Contemporary Educational Psychology, 13*(4), 348-357. http://dx.doi.org/10.1016/0361-476X(88)90034-3

Moreno-León, J., Robles, G., & Román-González, M. (2016). Code to learn: Where does it belong in the k-12 curriculum? *Journal of Information Technology Education, 15*, 283-303. Retrieved from: http://www.informingscience.org/Publications/3521

National Research Council. (2012). *A framework for K–12 science education: Practices, crosscutting concepts, and core ideas*. Washington, DC: National Academies Press.

Palumbo, D., & Michael Reed, W. (1991). The effect of BASIC programming language instruction on high school students' problem solving ability and computer anxiety. *Journal of Research on Computing in Education, 23*(3), 343-372. http://dx.doi.org/10.1080/08886504.1991.10781967

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*. New York, NY: Basic Books.

Pea, R. D., & Kurland, D. M., (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*(2), 137-168. https://doi.org/10.1016/0732-118X(84)90018-7

Pellas, N. & Peroutseas, E., (2017). Leveraging Scratch4SL and Second Life to motivate high school students' participation in introductory programming courses: Findings from a case study. *New Review of Hypermedia and Multimedia, 23*(1), 51-79. https://doi.org/10.1080/13614568.2016.1152314

Pinkston, G. (2015). Forward 50, teaching coding to ages 4-12: Programming in the elementary school. *Global Science and Technology Forum*, 34-39. http://dx.doi.org.helicon.vuw.ac.nz/10.5176/2251-1814_EeL15.11

Psycharis, S., & Kallia, M., (2017). The effects of computer programming on high school students' reasoning skills, and self-efficacy and problem solving in Mathematics. *Instructional Science*. 45. 583–602. https://doi.org/10.1007/s11251-017-9421-5

Resnick, M. (1994). *Turtles, termites, and traffic jams: Explorations in massively parallel microworlds*. Cambridge, Mass.: MIT Press.

Resnick, M. (2002). *Rethinking learning in the digital age. The Global Information Technology Report: Readiness for the Networked World*. Massachusetts. Oxford University Press.

Resnick, M., & Siegel, D. (2015). A different approach to coding. *International Journal of People-Oriented Programming*, 4(1), 1-4.

Resnick, M., Flanagan, M., Kelleher, C., MacLaurin, M., Ohshima, Y., Perlin, K., & Torres, R., (2009). Growing up programming: Democratizing the creation of dynamic, interactive media. In *Extended Abstracts on Human Factors in Computing Systems*. (pp. 3293-3296). New York: CHI. https://doi.org/10.1145/1520340.1520472

Sáez-López, J-M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education, 97*, 129-141. https://doi.org/10.1016/j.compedu.2016.03.003

Soloway, E., Lochhead, J., and Clement, J. (1982). Does computer programming enhance problem solving ability? Some positive evidence on algebra word problems. *Computer Literacy: Issues and directions for 1985*, 171-201. https://doi.org/10.1016/B978-0-12-634960-3.50023-3

Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. (2008). Bloom's taxonomy for CS assessment. *Proceedings of the tenth conference on Australasian computing education* (pp. 155-161). Wollongong, NSW: Australian Computer Society.

Te Kete Ipurangi (2017). *Technology in the New Zealand curriculum.* Retrieved from: http://nzcurriculum.tki.org.nz/The-New-Zealand-Curriculum/Technology/Progress-outcomes#collapsible2

Tondeur, J., Van Braak, J., Sang, G., Voogt, J., Fisser, P., & Ottenbreit-Leftwich, A. (2012). Preparing pre-service teachers to integrate technology in education: A synthesis of qualitative evidence. *Computers & Education*, 59(1), 134-144. https://doi.org/10.1016/j.compedu.2011.10.009

Webb, M., Davis, N., Bell, T., Katz, Y.J., Reynolds, N., Chambers, D.P., & Syslo, M.M. (2017). Computer Science in K-12 School Curricula of the 2lst Century: Why, What and When? *Education and Information Technologies, 22*(2), 445-468. https://doi.org/10.1007/s10639-016-9493-x

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Xia, B. S., (2017). An in-depth analysis of teaching themes and the quality of teaching in Higher Education: Evidence from the programming education environments. *International Journal of Teaching and Learning in Higher Education, 29*(2), 245-254.

Appendix 1

Sampling Frame of included articles

First Author	Year	Journal	Context of study	Topic	Method	Participants	Findings.
Bernardo	1994	Journal of Research on Computing in Education	USA	The transfer effects of computer programming on mathematical modelling, procedural comprehension, and verbal problem solution	Teacher-made pretests and posttests covering mathematical modelling, procedural comprehension, and verbal problem solution.	129 senior school students consisting of 2 control groups	Students learning BASIC had significant improvement in verbal problem solving that the control group and those studying computer literacy did not have. Gender did not make a difference. No significant difference in procedural comprehension or mathematical modelling between the 3 groups.
Falloon	2016	Journal of Computer Assisted Learning	New Zealand	Strategies used to plan for and integrate coding into a primary school numeracy programme and the thinking skills students use while completing their coding tasks.	Interviews, document analysis, iPad display, audio capture, recordings of students on screen activity and associated oral interaction.	36 primary school students	The way that coding is taught can lead to a range of generic skills or key competencies being developed.
Fessakis	2013	Computers & Education	Greece	The capability of 5 and 6 years olds in solving problems using computer programming.	Observation of video recordings of the learning activities, analysis of teacher's interview and researcher's notes.	10 primary school students 5-6 yr olds	The teacher was key in facilitating learning. Children were problem-solving and collaborating in their learning activities.

Hayes	2016	British Journal of Educational Psychology	Ireland	Comparing the effects of relational training and computer coding on intellectual potential in primary school students.	Pre-and post-assessment of standardised tests of IQ and academic performance.	28 primary school students	Scratch group improved less on the skills being measured than the other group.
Kalelioğlu	2015	Computers in Human Behavior	Turkey	The effect of teaching using the code.org site on reflective thinking skills towards problem solving and whether there is a gender difference in terms of students' reflective thinking skills towards problem solving.	Pre-test/post-test comparison of reflective thinking skills and focus group interviews.	32 primary school students	The test results showed no significant change in reflective thinking skills within problem solving. Qualitative data indicates that students were collaborating and self-managing to solve problems.
Kalelioğlu & Gülbahar	2014	Informatics in Education	Turkey	The effect of teaching scratch on primary school children's problem solving ability.	Pre-test/post-test comparison of problem solving inventory.	49 primary school students in 5^{th} grade.	No significant improvement in problem solving or reflective skills, but positive improvement in attitude to learning.
Miller	1988	Contemporary Educational Psychology	USA	Effects of LOGO Computer Programming Experience on Problem Solving and Spatial Relations Ability	Two computer programs were used to assess problem-solving differences. Spatial relations ability was measured by subtests of the California Test of Mental Maturity (CTMM)	174 primary school students from 2 different schools (one as a	Those learning LOGO did better on one of three spatial relations test and both of the problem solving tests. The problem-solving tests were carried out on the computer..

					and Primary Mental Abilities (PMA). General academic ability was also assessed in comparison from previous years' grades as a covariate.	control group)	
Palumbo	1991	Journal of Research on Computing in Education	USA	To investigate primarily the effect of programming language instruction on problem solving skills.	Pre-test and post-test comparisons on problem-solving skills using a variety of statistical measures.	22 high school students (11 as a control group)	Students learning BASIC had significant improvement in problem solving that those studying computer literacy did not have.
Psycharis	2017	Instructional Science	Greece	To investigate whether programming has an impact on reasoning skills, problem solving and self-efficacy in Mathematics.	Quasi experimental, pre and posttests using Cornell reasoning test, MSQL and a problem adapted from the Greek national exam board. Compared students learning programming and those not.	66 high school students.	Significant difference in reasoning skills for those participating in computer programming. No significant difference in problem-solving skills between the two groups.
Sáez-López,	2016	Computers & Education	Spain	To assess the use of a Visual Programming Language using Scratch in an integrated curriculum, analysing the outcomes and attitudes of primary school students.	Pre-and post-tests assessing quantitative data, student observations and questionnaires analysing qualitative data.	107 primary school students from 5 different schools (over two-years) plus a control group of 32 students	The students made significant gains in understanding programming. Active learning was measured and scored highly as did the learning of Art history course content.

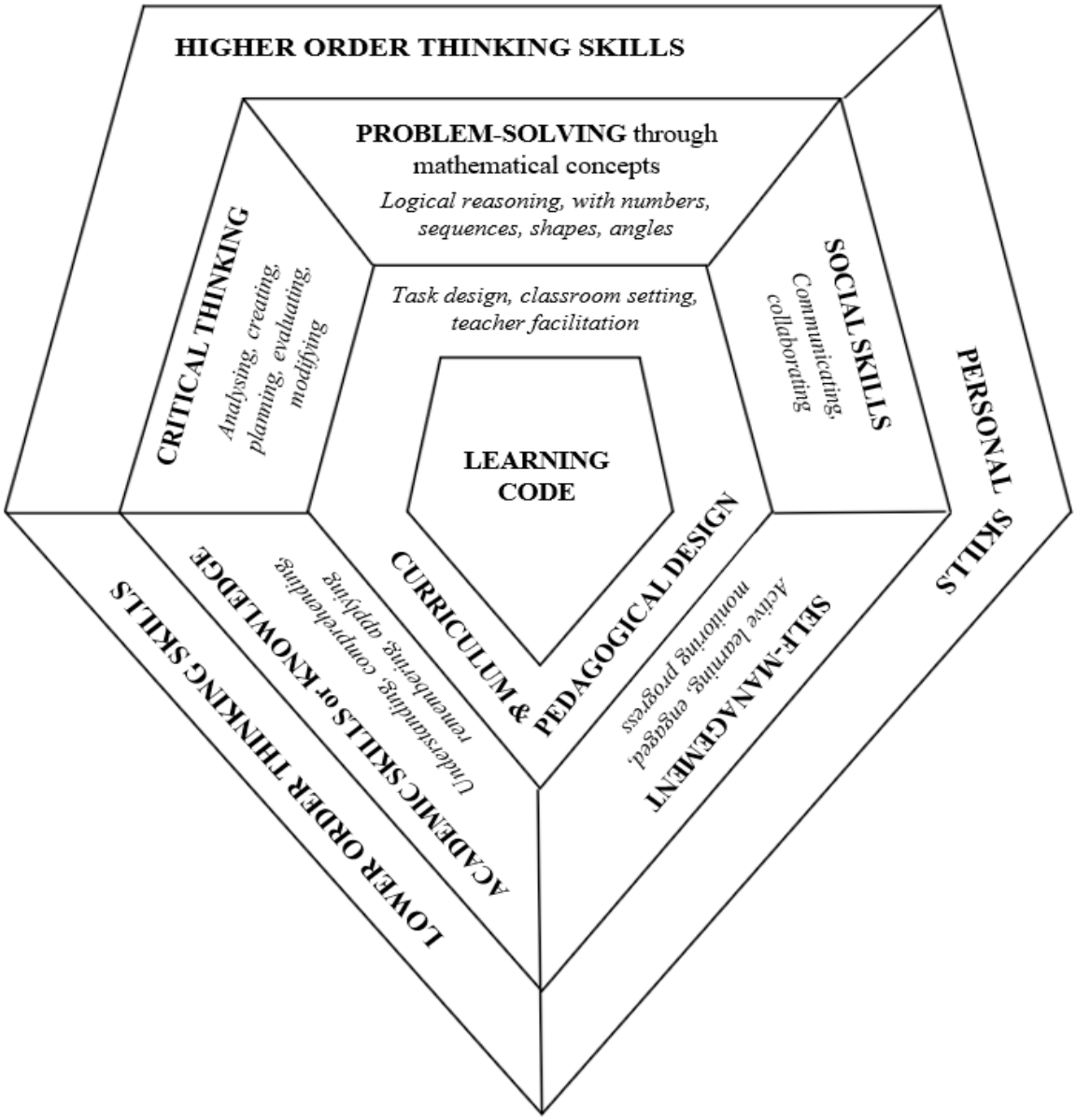

Figure 1. *Model depicting the influence of coding on educational outcomes.*