



A Model Suggested for Programming Teaching: Programming in Seven Steps

Kürşat Ali Erümit ¹, Hasan Karal ², Güven Şahin ³, Dilara Arzugül Aksoy ⁴,
Ayşegül Aksoy Gencan ⁵, Ali İhsan Benzer ⁶

Abstract

There are a number of studies in the literature revealing that programming instruction has positive effects on development of different cognitive skills. However, there are scarcely any studies suggesting pedagogical approaches to how programming should be done. Therefore, this study focused on how to do programming instruction that target developing students' cognitive skills. In this study, it was aimed to suggest a model for teaching in order to develop various cognitive skills of students. Within the scope of the study, a programming instruction model was created, and it was called Programming in Seven Steps (PSS) model. As research design, special case study was used from qualitative research methods. The research team consisted of two groups: design team (1 faculty member from the department of Computer Education and Instructional Technologies (CEIT), 2 master's degree students, and 2 PhD students) and a Design Evaluation Team (DET) (10 IT and Software teachers who work in secondary schools affiliated to the Ministry of National Education) in the process of constructing the PSS model. During the research, the documents and interviews with the design evaluation team were analyzed with the purpose of determining the steps of the PSS model. The study elaborates the process followed to create the model, the features of the proposed model, and interviews with the design evaluation team. This article details the overall path to designing of the model, the features of the model, and DET interviews. It is aimed that the model introduced in this study will become a guide for educators who want to teach programming at secondary school level.

Keywords

Programming teaching
Problem solving
Algorithmic thinking
PSS teaching model
Lesson plans
Secondary school
Coding

Article Info

Received: 12.29.2017
Accepted: 09.25.2018
Online Published: 12.18.2018

DOI: 10.15390/EB.2018.7678

¹ Trabzon University, Fatih Education Faculty, Computer and Instructional Technology, Turkey, kursaterumit@gmail.com

² Trabzon University, Fatih Education Faculty, Computer and Instructional Technology, Turkey, karalhasan@gmail.com

³ Trabzon University, Fatih Education Faculty, Computer and Instructional Technology, Turkey, guvnnshahinn@gmail.com

⁴ Trabzon University, Fatih Education Faculty, Computer and Instructional Technology, Turkey, dilaaarzugulaksoy@gmail.com

⁵ Trabzon University, Fatih Education Faculty, Computer and Instructional Technology, Turkey, aysegul.aksoy.61@gmail.com

⁶ Trabzon University, Fatih Education Faculty, Computer and Instructional Technology, Turkey, aibenzer@gmail.com

Introduction

The developments in science and technology cause skills expected from individuals to differentiate. Trilling and Fadel (2009) define the 21st century skills as verbal and written communication, critical thinking and problem solving, professionalism and work ethics, collaboration and teamwork, technology enforcement, leadership and project management, and working in various teams. ISTE (2016) emphasized that students should have certain standards for learning effectively. These standards were classified as creativity and innovation, communication and collaboration, research and information flow, critical thinking, problem solving and deciding, digital citizenship, and use of technology. Besides these, computational thinking skill is also regarded necessary.

Denoted as a broad concept covering skills of problem solving, system design, and understanding human behaviors (Wing, 2006); computational thinking also refers to the use of computers to develop cognitive processes and problem solving skills (Sanford & Naidu, 2016).

The ability of computational thinking, which involves using algorithms to solve problems effectively (Atmatzidou & Demetriadis, 2016; Choi, Lee, & Lee, 2016), is at the center of computer science with this feature. For this reason, besides understanding the algorithm design principles, a learner's solving a problem with appropriate algorithms in reference to these principles can improve the learner's computational thinking skill (Choi et al., 2016). Designing appropriate algorithms for solving a problem is closely related to algorithmic thinking skills. Algorithmic thinking ability is expressed as ordering of actions by an individual by thinking creatively and logically (Ziatdinov & Musa, 2012) and is a key skill that ensures development of other cognitive skills directly related to programming.

According to Futschek (2006), algorithmic thinking consists of various sub-skills related to comprehendin and configuration. These sub-skills are the ability to analyze a given problem, to fully express a problem, to produce a strategy for a given problem, skill of constructing an algorithm for a given problem using the strategies, the ability to think in all possible special and normal cases, and the skill of increasing the efficiency of an algorithm. Futschek (2006) argues that algorithmic thinking can also be developed with applications such as games (Maze) and sorting (Parallel Sorting) in which different strategies can be applied independently of the computer environment. Therefore, students can test an algorithm which they develop for solving a problem, before computerizing it.

One of the ways to develop algorithmic thinking ability is programming instruction. In the literature, it is stated that programming develops various skills of learners. It is known that:

- Programming develops problem solving skills of students, (Bergersen & Gustafsson, 2011; Brown et al.; Kalelioğlu & Gülbahar, 2014; Lai & Lai, 2012; Lai & Yang, 2011),
- It has positive effects on cognitive learning (Clements & Sarama, 2003; Crescenzi, Malizia, Verri, Diaz, & Aedo, 2012; Grover & Pea, 2013; Utting, Cooper, Kölling, Maloney, & Resnick 2010),
- It enhances high-level thinking skills of students (Kafai & Burke, 2014; Shih, 2014),
- It increases motivation (Akpınar & Altun, 2014),
- It improves creative thinking ability (Fesakis & Serafeim, 2009; Kobsiripat, 2015).

Besides developing cognitive skills of students, programming requires high-level thinking abilities (Law, Lee, & Yu, 2010). For this reason, it is stated that the programming instruction process is quite difficult (Helminen & Malmi, 2010) and students have a lower level of success in this lesson (Robins, Rountree, & Rountree, 2003). These difficulties occur during learning of the basic programming concepts such as structure of program (Lahtinen, Ala-Mutka, & Jarvinen, 2005), cycles (Ginat, 2004), and algorithm structure (Seppala, Malmi, & Korhonen, 2006). At the same time, students may have difficulties in the programming course because of the teaching method used. In many studies, it is discussed how programming instruction should be despite the mentioned difficulties (Coull & Duncan, 2011; Lahtinen et al., 2005).

In order to increase success in programming and facilitate understanding, it is initially required to teach the logic of algorithm to students (Ala-Mutka, 2004). Due to this need, interesting and amusing visual programming languages have been developed to facilitate learning for beginners (Schwartz, Stagner, & Morrison, 2006). In text-based programming, code-sequencing emerges as one of the most troublesome components for students (Özmen & Altun, 2014). Code blocks are offered to facilitate understanding and implementing by overcoming this challenge in visual programming (Wilson & Moffat, 2010). Sáez López, González, and Cano (2016) pointed out that working on projects with the aid of visual programs increases motivation and willingness of students. Most studies reported that the use of visual programming accelerates comprehending (Naharro-Berrocal, Pareja-Flores, Urquiza-Fuentes, & Velazquez-Iturbide, 2002). For beginner-level learners of programming; programs such as Scratch (Malan & Leitner, 2007; Wu, Chang, & He, 2010), kodu (Stolee & Fristoe, 2011), StarLogo (Klopfer & Yoon, 2005), and Alice (Kelleher, Pausch, & Kiesler, 2007) are suggested.

According to Robins et al., (2003), teaching of programming must focus on not only learning properties of the new language but also use of such properties in different situations, particularly on the idea underlying the basic program design. Linn and Dalbey (1989) offers the “the chain of cognitive accomplishments” for programming instruction. This chain starts with language features. The second circle consists of designing skills covering templates, planning, testing, and reformulating. The third circle is a problem solving skills that involves adapting knowledge and strategies to a new programming language.

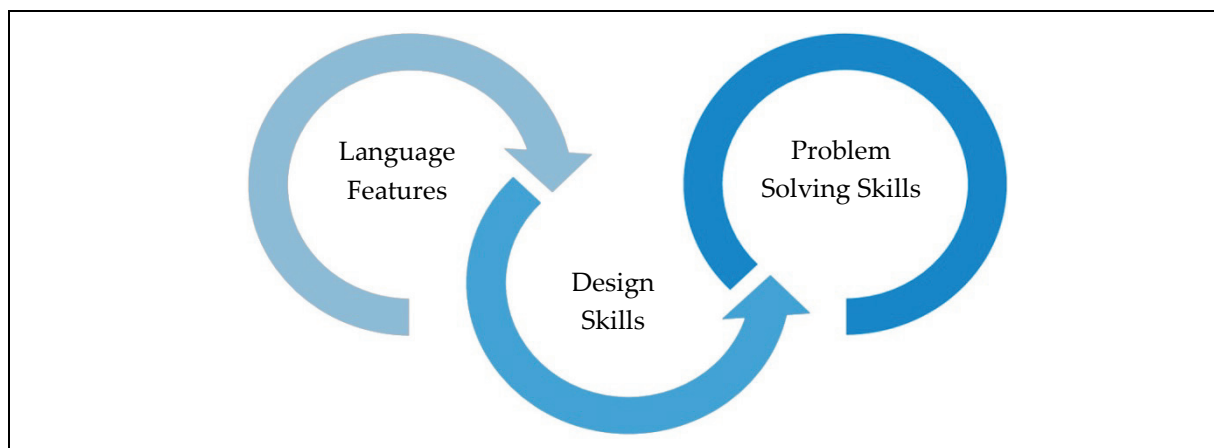


Figure 1. Cognitive Competences Necessary for Programming Teaching (Linn & Dalbey, 1989)

The chain in Figure 1 symbolizes introduction to programming. In a setting of teaching in this mode, an instructor should clearly express the control structures, data structures, program design, and the problem area. Otherwise, it will not be possible to form a strong background (Winslow, 1996). During instruction process, student motivation can be increased by giving visual and animation-based programming activities. At that point, it is considered important that the instructor explains his visual-based activity in reference to the basis concepts of programming (Kurland, Pea, Clement, & Mawby 1989). Spohrer and Soloway (1989) also made some recommendations regarding programming instruction:

- Use graphical languages for flow control
- Use a simple machinery model
- Before determining variables and constant names, specify a simple and consistent naming rule
- Provide design samples based on spatial metaphors
- Gradually decrease the support to students

During the programming teaching process, students should be given continuous and rapid feedback. In addition to this, it is recommended to allow students for their own learning (Linn & Dalbey, 1989) and collaborative works (Van Gorp & Grissom, 2001; Williams, Wiebe, Yang, Ferzli, & Miller, 2002).

Because there are scarcely any studies suggesting pedagogical approaches to how programming should be done, there arise differences in trainings run for the same age group in educational institutions. These differences occur in the form of differentiation of activities and followed steps for programming instruction. In the case of an instruction program, which is aimed to ensure development of students' cognitive skills, this situation could lead to;

1. Turning of the lesson into an animation creation and leisure time field rather than developing cognitive skills,
2. Dragging away of students from programming due to abundance of animation-like activities in the lesson
3. Losing of probable positive effects of the IT and software course on other courses
4. Losing importance of the course for students, parents, and authorities
5. Failure of the instruction program to reach the goal.

One of the main reasons for this situation is that animation-related activities that teachers try to teach in schools can be realized by students by themselves due to the easy access to and learning of Web 2.0 technologies (Karaman, Yıldırım, & Kaban, 2008; Kam & Katerattanakul, 2014). Neglecting effects on development of cognitive skills may cause students to perceive programming instruction equal to training on animation preparation (Lee, 2011; Scaffidi & Chambers, 2011) and underestimate the course and thus banalize the training in schools gradually being questioned (Kalelioğlu & Gülbahar, 2014; Kukul & Gökçearsan, 2014; Oluk & Saltan, 2015; Yükseltürk, Altok, & Üçgöl, 2016).

In the 21st century, programming teaching has an important place in the development of the skills necessary for individuals to be productive and active. For this reason, programming teaching has started to take its place in the curriculum of many world countries (Balanskat & Engelhardt, 2015). It is seen that countries integrate programming teaching into curricula not only in order to develop programming skills but also to develop logical thinking and problem solving skills of learners (Balanskat & Engelhardt, 2015). However, programming teaching is seen as a difficult process to perceive for students (Porter & Calder, 2004). This is because the programming tools and programming languages have a structure hard to understand, the syntactic rules for the programming language are difficult, traditional methods are used in programming teaching, and students are not competent in algorithmic thinking (Byrne & Lyons, 2001; Futschek, 2006). Departing from this, the aim of this study is to propose a teaching model that can be used in the programming teaching process for the development of algorithmic thinking, problem solving and programming skills at the secondary school level.

Method

Research Model

In this study; it is aimed to suggest a model for programming instruction to develop problem solving, algorithmic thinking, and programming skills at the secondary school level and opinions of the Design Evaluation Team (DET), composed of teachers, were taken to this end. In this direction, first of all, appropriate steps for programming instruction were searched and the steps of the PSS model were determined through document analysis. Then, the DET were interviewed by using semi-structured interview forms in order to check the cognitive level, duration, and applicability for overcrowded classrooms of the lesson plans prepared in this study. Then the plans were revised accordingly. As research design, special case study was selected amongst qualitative research methods. Special case study is defined as a research pattern by which qualitative data collection methods such as observation, interview, and document analysis are used to explore perceptions and facts in a realistic and holistic manner in their natural setting (Yin, 2017). The pattern was preferred also because it allows flexibility to researcher during design and implementation of the study (Silverman, 2013).

In this study, the special case design was selected for

- Analyzing the key words and document analyses on databases for the purpose of identifying the PSS steps,
- In-depth analysis of the lesson plans, activities, and PSS model steps with content analysis following interviews with certain DET members,
- Better revealing feelings, thoughts, and emotions by means of qualitative methods, and
- Describing the findings in a qualitative manner.

Research Group

The study employed two study groups at stages of formation of the PSS model and evaluation of the lesson plans prepared according to the created model. The design team (DT) consisted of 5 experts (1 CEIT faculty member, 2 graduate students, 2 doctoral students) and the 10 design evaluation team (DET) consisted of another 10 people (10 teachers of IT and Software Course employed by the Ministry of National Education). The personal information about the TE is given in Table 1 and DET in Table 2.

Table 1. Design Team

Team Member	Title/ Level of Education	Experience		
		Programming	Programming Teaching	Bachelor Degree
C1	Assistant Professor	21 years	14 years	Computer Teacher
SP1	Lecturer-PhD Student	17 years	14 years	Computer Teacher
SP2	PhD Student	8 years		Computer Education and Instructional Technology Teacher
SM1	Graduate Student	6 year		Computer Education and Instructional Technology Teacher
SM2	Graduate Student	6 years		Computer Education and Instructional Technology Teacher

The design team indicated in Table 1 took a role in devising the PSS model as well as the lesson plans and the activities in accordance with the model. The study was conducted during the 2016-2017 autumn semester for 6 hours in 16 weeks. The teachers who currently teach in secondary schools affiliated with the Ministry of National Education (Design Evaluation Eeam) were in charge of

evaluating the draft model, lesson plans, and activities as seen in Table 2. Activities and lesson plans were prepared for each topic by the DT so that teachers can see the concrete steps of the PSS model and thus make a more realistic assessment. During interviews with the DET; firstly, the steps of the PSS model were explained and the corresponding actions and activities were shown in the lesson plans, and then the opinions of the teachers were taken. All of the teachers have a bachelor's degree in Computer Education and Instructional Technology.

Table 2. Design Evaluation Team

Teacher	Experience	
	Teaching	Programming Teaching
Ö1	13 years	4 years
Ö2	10 years	4 years
Ö3	10 years	4 years
Ö4	6 years	4 years
Ö5	6 years	4 years
Ö6	14 years	3 years
Ö7	12 years	2 years
Ö8	10 years	2 years
Ö9	10 years	2 years
Ö10	10 years	1 years

The activities and lesson plans prepared by the DT were checked regarding conformity with the cognitive level, applicability in large classrooms, and duration by means of interviews with the DET. Subsequently; the model, the lesson plans, and the activities were developed in line with the comments.

The Course of Research

In line with the study aim, the research process was carried out following the ADDIE design model. ADDIE is a teaching design model consisting of the initials of the words Analyze, Design, Develop, Implement, and Evaluate, which involves five steps. The ADDIE model was preferred in this study as it is a basic model that can be suitable for any learning and includes components of other instructional design models. The course of research following the model is depicted in Figure 2 with detailed actions at each stage.

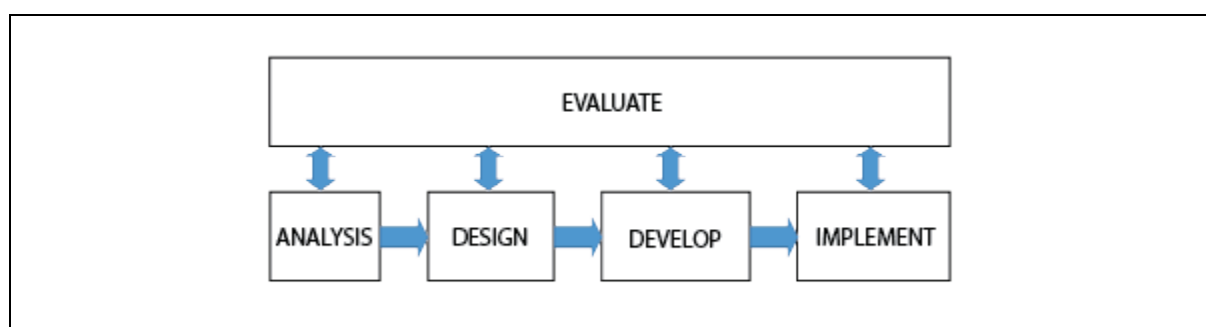


Figure 2. The Course of Research According to the ADDIE Design Model

(1) Analysis

The stage of analysis was performed in three steps as problem analysis, task analysis, and instructional analysis. In problem analysis, the DT reviewed the literature. It was found striking that there are only few studies suggesting a pedagogical approach to teaching programming at the secondary school level. Moreover, it was seen that there is no common approach to how programming can be done. For this reason, the present study was started from the question 'How should programming

instruction be done?' In task analysis, the result obtained from the problem analysis led to the task as 'proposing a model for programming instruction that includes algorithmic thinking and problem solving skills competencies at the secondary school level'. Finally, within the scope of instructional analysis; attainments under "Concepts and Approaches to Problem Solving", "Let's Recognize the Programming Environment", "Variables", "Conditionals", and "Cycles" were retrieved taken from the MEB (2017) Curriculum for IT and Software Course.

(2) Design: Identification of the Instructional Steps

The document analysis performed for identification of instructional steps followed the suggested by Yıldırım and Şimşek (2011) as access to documents, confirming the authenticity, comprehension of documents, analyzing data, and using data.

Access to Documents

To start with, literature review was done on algorithmic thinking skill as the basis of programming instruction by using key words such as *Algorithmic Thinking, Curriculum, Primary and Secondary education, Problem Solving, Computational Thinking, and Deeper Learning* on Google Scholar, Science Direct, and Eric. The review gave 15 studies, 6 of which were chosen as they discuss levels and steps that can be utilized in the instruction process. The samples utilizing certain pedagogical approaches to implementation of activities (binary coding, drama, etc.) but not dealing the whole lesson and those not detailing the course of study were discarded. Papers were used as references in this study in order to be able to set the limits of the clearly considering applicability of the study. The examples scrutinized for this purpose are listed in Table 3.

Table 3. Examined Documents

Types of Documents	Examined Resources
Papers	Zsakó and Szlávi (2012), ICT Competences: Algorithmic thinking
	Vasconcelos (2007), Basic Strategy for Algorithmic Problem Solving
	Futschek (2006), Algorithmic thinking: The key for understanding computer science
	Committee on Logic Education (2008)
	Szántó (2002) as cited in Zsakó and Szlávi, 2012, p. 55
	Garner (2003), Learning resources and tools to aid novices learn programming

Confirming the Authenticity

It is important to confirm the authentic nature of the documents reached for validity of the study. In this context, the search was conducted with the keywords mentioned above in the Google Scholar, Science Direct and Eric databases, and the authenticity of the published articles was confirmed by checking the journals. The resources were checked particularly for index and existence of editorial board.

Comprehension of Documents

At this stage, the documents were cross-checked systematically by examining the levels and steps that could be used in the process of instruction. The papers were subjected to content analysis to elicit the themes by the DT.

Analyzing Data

The themes derived in the previous stage are shown in Table 4.

Table 4. Comparison of the Themes Identified for the Instructional Steps

Studies	Understand the problem	Devise a plan	Comparing plans	Devise an algorithm	Code the algorithm	Identify and correct an error in a different code	Prepare and Code New Algorithms
Zsakó and Szlávi (2012)	✓	✓	✓	✓	✓	✓	✓
Vasconcelos (2007)	✓	✓		✓	✓		✓
Futschek (2006)	✓	✓		✓	✓		
Committee on Logic Education (2008)	✓		✓	✓			
Szántó (2002)				✓	✓		
Garner (2003)	✓			✓	✓	✓	

The analysis process was completed with the simultaneous work by the DT through consensus. The themes elicited by the team were examined at meetings and it was seen that the themes intersect in the factors of Understand the Problem, Devise a Plan, Compare Plans, Devise an Algorithm, Code the Algorithm, Identify and Correct an Error in a Different Code, and Prepare New Algorithms and Code. Information on the identification of the themes was given under the heading "Validity and reliability of the study".

Using the Data

The results and data obtained from document analysis are discussed in under findings and results. The steps included in this study following examination of the levels and steps that can be used as instructional steps are detailed under the study findings, while steps of the created PSS model are given under the results.

Identifying the Lesson Plan Evaluation Criteria

By using the lesson plan evaluation criteria attached as Annex-1, views of 150 teachers were obtained as a response to the question "What do you think should be considered in preparing a lesson plan?" As a result of analysis of the responses by the Design Team, 3 common criteria were identified as "conformity with the cognitive level", "applicability in large classrooms", and "conformity with duration". Table 5 displays distribution of these criteria according to the DT views.

Table 5. Distribution of the Criteria (n=150)

	f	%
Conformity with the cognitive level	96	64
Conformity with duration	64	42,66
Applicability in large classrooms	86	57,33

It is seen in Table 5 that the teachers stated that activities to be designed must comply with the cognitive level as the most important consideration while preparing a lesson plan. As the second most important item, they stressed that these activities should also be suitable for being applied in overcrowded classrooms. Another point that they draw attention to is that the activities must be tailored to the duration of the lesson.

(3) Refinement: Preparation of Activities and Lesson Plans by the Design Team

Taking into consideration the MoNE curriculum, the DT began to develop the lesson plans and activities to be covered. Activities to be developed are of crucial importance for feasible steps within the SSP model. In the development of activities, compliance with students' cognitive levels, their applicability in overcrowded classrooms, suitability for the duration of the class, and adaptability to the computer environment were taken into consideration. Another point to be considered when determining the activities is that these activities must be designed analogously. Analogy covers the process of understanding an unknown event by means of a known event, comparing and establishing relations between the two events (Çıray & Erişti, 2014). Analogy as a method has a crucial role to play in making students active participants and encouraging their inquisitive and creative skills (Yuretich, Khan, Leckie, & Clement, 2001). In this study, the activities were created in accordance with analogies and allowing students to express themselves freely during the activity. Also, attention was paid to including examples from daily life. This type of activities improves clarity of topics by simplifying topics of programming which are difficult to be comprehended by students. In addition, it was considered important that the activities in the lesson plans involve a problem situation with multiple strategies for solution of the problem. The DT members presented their individual works in group meetings for evaluations of the group members. Decisions taken regarding the individual works during the meetings were noted and the plans were re-evaluated by the other group members. As a result, the final version of plans was approved in the light of the design team's comments.

(4) Implementing: Submitting the Plans to Teacher Views

After drafting as commented by the DT, the lesson plans were evaluated against the lesson plan evaluation criteria above through semi-structured interviews with the DET. The interview questions are attached as Annex-2.

(5) Evaluation: Updating the Plans

The lesson plans were updated by taking into consideration the data obtained through the DET interviews. In order to finalize the lesson plans, all teacher interviews and updates were completed in about 6 weeks after the 10-week process and the plans were made ready for implementation. Since this study focuses on the PSS model and relevant discussions, the process of developing appropriate lesson plans are not described here. Detailed information and sample lesson plans are available in Şahin (2018).

Validity and Reliability of the Study

The qualitative data, which include the participants' expressions, the process of combining, simplifying, and interpreting of the researcher's observation and reading (Merriam & Tisdell, 2015), were analyzed with content analysis. Once the semi-structured interviews with teachers and students had been recorded, they were analyzed and data obtained through analysis of the recordings were transferred onto the computer. Later, as Creswell (2013) stated, the data were edited, coded by themes, and finally the themes were supported with direct quotations. Measures were taken to prevent factors that may threaten validity and reliability throughout the research process. For example, the participants were informed in advance of the interviews, participation was scheduled completely on voluntary basis, and participant information was kept confidential. Voice recordings were made during interviews, the researcher kept notes, and every attempt was made to record the data without missing anything. All of the collected data were carefully stored so that the results could be verified. In the analysis phase of the data, three researchers who are specialized in Computer Education and Instructional Technology took part and then consensus was sought among the analysis results. Each of the researchers determined the themes and the areas concerned separately. The themes initially determined by the researchers were found to be consistent by 92%. The researchers then came together and reached a consensus on the themes and related areas. Where necessary, the data were confirmed by re-checking with the participant. Finally, the research report was drawn up with a detailed description of the whole process, and a transferable research was obtained. The researchers themselves were actively involved in analyzing and reporting the data.

Results

In order to determine the instructional steps in this study, the data obtained from the literature review by the DT were assessed by the whole team and 7 steps were reached through consensus. In this process; first, the relationship between the steps and the cognitive skills was checked. Next, suitability of the steps for students' cognitive levels and applicability in overcrowded classrooms were discussed. In the findings section; the findings from document analysis and DET views are given which helped identify the steps of the PSS model.

Identifying the Steps of the PSS Model

The PSS model was designed to realize programming instruction to develop algorithmic thinking, problem solving and programming skills of secondary schoolers. The studies satisfying the relevant criteria were reached through document analysis and findings are shown in Figure 3. The steps shown in Figure 3 are described as algorithmic thinking steps in the relevant studies.

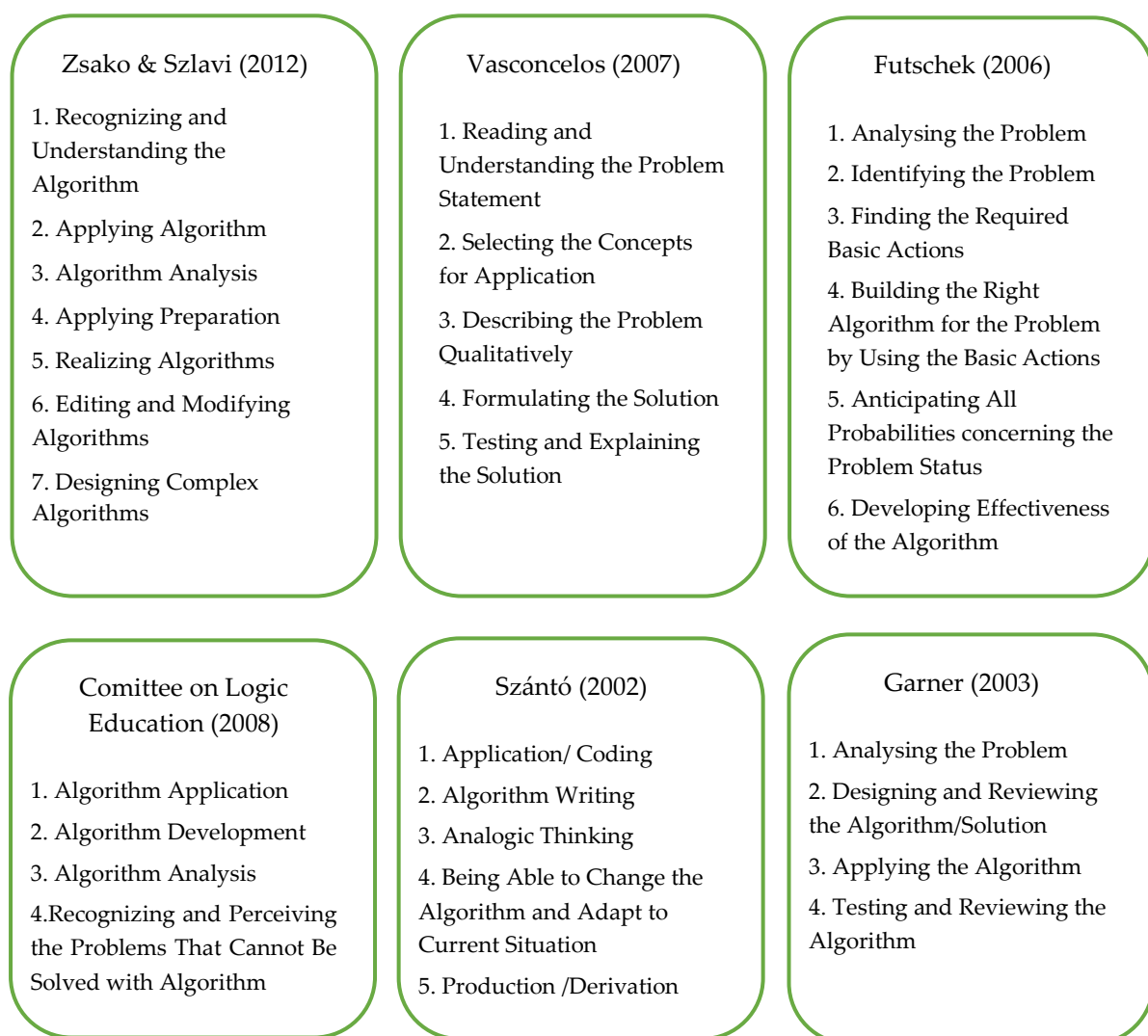


Figure 3. Stages of Algorithmic Thinking

Overall evaluations by TD regarding the approaches in Figure 3 are indicated as follows:

It is seen that the stages designed by Zsakó & Szlávi (2012) are targeted to manage the process of algorithmic thinking by grouping it. For example, the step of coding with the help of the programming language is introduced in the fifth step, whereas the others introduce the same stage

earlier. In this study, it was stated that each stage was built on the previous stage, and the relations of each stage to cognitive skills were underlined. It was stated that advanced stages require high-level thinking skills.

In his five-stage plan, Vasconcelos (2007) seems to place more focus on understanding and explaining the problem. On the other hand, not much emphasis is placed on editing and improving algorithms and designing complex algorithms. Still, it is important to elaborate what each step represents and what needs to be done then.

In Futschek's (2006) six-stage algorithmic thinking process, more stress is placed on understanding and explaining of the problem. However, the steps of creating complex algorithms receive less emphasis and the steps that make up the stages do not get much explanation.

Committee on Logic Education (2008) groups together the steps that describe the algorithmic thinking process and gives less place to the steps of analyzing and correcting errors, and designing complex algorithms particularly after testing the algorithm. It would be more appropriate to discuss the stages in more detail since this instruction is planned for the secondary education level.

Szántó (2002) as cited in Zsakó and Szlávi (2012) groups distinct steps in algorithmic thinking process, thus reducing the total number of steps.

Likewise, Garner (2003) reduces the number of steps by combining the steps designed. The steps that make up the stages are not discussed in detail.

In the study, the stages defined as levels of algorithmic thinking skill were then checked against the criteria of cognitive skills coverage and applicability in overcrowded classes, and the instructional steps were identified consequently.

Views of the DET concerning the PSS Model

Following the semi-structured interviews with the DET, the findings related to the applications at steps of the PSS model were tabulated in Table 6 under the title of Compliance Check of the Model.

Table 6. Compliance Check of the Model

Steps	Appropriate (n=10)	Updateable (n=10)		Topics	Recommendations (n=10)
		Compliant to the level	Applicability		
Understand the Problem	8	2	-	Conditionals	"I think it is an appropriate step. They already do such calculations in mathematics class so they won't have difficulty. Suitable for their level. But I suggest that what you think is nice but can it be simplified? I think it'd be easier to implement if you can do it in a simpler framework of events because it can be challenging."(Ö5)
				Cycles	"The step is absolutely necessary and logical. The activity was a bit difficult for me, considering students' level. If the child is struggling at first, there can be an impression on him that s/he can not succeed, so the activity may be simpler."(Ö7)
Devise a Plan	8	2	-	Conditionals	"It seems difficult for students to be able to overcome it individually. If group work is done, at least those who can not overcome will benefit from it." (Ö6) The idea is very nice, but the activity could possibly be simplified by shortening it. It seems a bit complicated in this way."(Ö7)
Compare the Strategies	9	-	1	Cycles	"If it is not a case of writing on the board but a different kind of writing practically or transferring in a different way, even 20 minutes will be too long, but it is a waste of time because it is the task of writing on the board."(Ö9)
Devise an Algorithm	5	-	5	Conditionals	"When we think about our own classes, only the space in front of the board exists and this may not be enough. In such a case, it would be more convenient if there were an alternative like going out of the class or using the garden or the corridor."(Ö5)
				Cycles	"Classroom environment can be preferred for PC-free activities."(Ö8)

				Variables	<p>“At the beginning of this application, we can create a variable for the children and make an example of using that variable in the calculation.”(Ö1)</p> <p>“In order to be able to do such coding, it is necessary to tell the children the mathematics of this work. Instead of starting with an application like this about variables, starting can be made with a much simpler calculation process.”(Ö3)</p> <p>“The first example we give can be difficult for the child to understand what s/he is doing. I think it can be started with an example that the child will do one single transaction.”(Ö8)</p>
Code the Algorithm	2	8	-	Cycles	<p>“Processes up to the computer activity have been described very well, but the first example in transition to the computer environment has been difficult. I think that it would be better to start at medium level in computer based projects.”(Ö2)</p> <p>“It can be started with a simpler example so that the child can discover it.”(Ö5)</p>
				Conditionals	<p>“I'd show something simpler if it were me. Directly thus and so. It may be healthier to start with a shorter example and then make it more difficult. This sounded very hard to me for the first example. The same example may be simpler.”(Ö8)</p>
Identify and correct an error in a different code	9	1	-	Conditionals	<p>“Before switching to the application for students at code editing stage, it could be more efficient if there were a lecture based on a structure using similar codes.”(Ö4)</p>
Prepare and Code New Algorithms	10	-	-		-

The DET evaluations of the lesson plans following the PSS model against the applicable criteria of cognitive level, duration, and applicability in crowded classrooms in Table 6 demonstrates that the steps of the SPP model are substantially appropriate. Teachers' evaluations of the steps are given below.

“Understand the Problem”

Variables- “I see fit for the sixth grade, so there are very descriptive sentences already in the problem, as far as I can see, there are no words that they will not understand as a term or as an expression, I see it quite understandable, conforming to their level.”(Ö9)

Conditionals- “Suitable. I think that a different topic attracts attention. Compliance is no problem, just a little complicated. Look at the picture to understand or trying to comment on the topic, all these might get confusing so it is necessary to go step by step regularly.”(Ö4)

Cycles- “It is a fun activity that attracts the attention of students and fit to their level.”(Ö4)

“Devise a Plan”

Variables- “Suitable because they always do that kind of transactions in maths lesson. Sixth graders will be able to easily comprehend and resolve.”(Ö3)

Conditionals- “I think it suits students' levels. Only while moving from one transfer point to the other transfer point, I think it is necessary to emphasize those transfer points a little more clearly or put forth them well on the picture while describing.”(Ö2)

Cycles- “Vehicle transits might distract children, apart from that, they can do it. I contemplate how will they reach the fastest? I think children will choose it. Our sixth grade students can do.”(Ö9)

“Compare the Strategies”

Variables- “15 mins is acceptable. For this item, it is not an item which will take too long to discuss strategies proposed by children, they are able to express their own strategies in a few minutes.”(Ö7)

Conditionals- “I think this duration, he'd talk if he were allowed two lessons' time. Because at that age they always are, but what is good here is that one can say what another can not think, or another group if they work in groups.”(Ö4)

Cycles- “20 minutes is enough to examine the strategy for the third item and to examine it after identifying the strategy 20 minutes. Sufficient time is allocated. Suitable.”(Ö10)

“Devise an Algorithm”

Table 7 reveals that 5 teachers found the step “Devise an Algorithm” good enough. The other 5 teachers stated that some of the drama games at that step are not practical for the size and physical facilities of classes. Some of the teachers' views regarding this step are given below:

Variables- “I do not think there will be much trouble in crowded classrooms because in this kind of activities, the class can draw the attention onto that completely. If balance is achieved without too much laxity, there will be no problem in practice in crowded classrooms.”(Ö7)

Conditionals- "In practice this animation seems very difficult to me. The first is to create physical space; the second seems to be difficult for children to make exchanges or the idea of transfer with these different people."(Ö3)

Cycles- "If it is a crowded class, creating an appropriate environment is a nuisance. When it is crowded, the student at the back will not be able to follow the activity exactly. There may be such problems."(Ö4)

"Code the Algorithm"

In Table 7, while computerized activities for "Code the Algorithm" were considered acceptable by 2 of the teachers, 8 members of the team thought they could be revised for higher compliance with cognitive levels. Some of the teachers' views in this regard are quoted below:

Variables- "Once they define the variables and understand which variable is what, they can do the definition but they may not be able to sort out adding, multiplication and so on with the variables."(Ö1)

Conditionals- "The examples are good but it is controversial whether they are for secondary school or upper level. The simpler, the better. It is good that the examples are related to everyday life."(Ö7)

Cycles- "This is quite like a year-end project. Generally speaking, it is difficult even to do the rotating movement."(Ö6)

Identify and Correct the Error in a Different Code

Table 7 reveals that computer-aided activities corresponding to "Identify and Correct the Error in a Different Code" were approved by 8 teachers, while 2 teachers expressed opinions for updating. Below are some of the teachers' views:

Variables- "10 minutes might be short to learn and correct the error. A good activity to find and correct the error in the algorithm but it may take a little longer to detect the error. Correction can be fast, but time may get shorter until detecting the error."(Ö7)

Conditionals- "A simple application that students can resolve. I think they can edit the bad code structure. Suitable for student level."(Ö6)

Cycles- "It can be done, pretty good. It's such a good thing because each one has got a mistake; it will control both the cat and the dog and if it goes step by step, there will be no trouble, and when it reads step by step, it will see the error already."(Ö9)

The DET made comments predominantly on the fourth and fifth steps of the PSS model. Although they found the model steps acceptable and essential, they mentioned updating for the sake of more efficient activities connected to these steps. Table 7 displays the considerations in preparing lesson plans in accordance with the PSS model from the DET's perspective.

Table 7. Considerations in Lesson Plans

PSS Model Steps	DET Recommendations
Understand the Problem	Before moving on to the main activity, you should start with a simpler activity about the topic or a simpler part of the main activity
Devise a Plan	-
Compare the Strategies	At this stage, the teacher must write on the board in order to help students put their different strategies and steps in an algorithm. In this way, students are given more time to develop their own strategies and compare with other students' strategies while spending less time for writing.
Devise an Algorithm	-
Code the Algorithm	Before the generated algorithm is coded in its entirety, it must be divided into segments to progressively go from simple to complex. Instructions that are not part of the lesson and that confuse students (moving the character or changing a costume, etc.) should be made available to the students. So, without distracting the students, they must focus only on the topic of the lesson.
Identify and Correct an Error in a Different Code	Students should be given as much time as possible to analyze the codes as required by the transactions in this step.
Prepare and Code New Algorithms	-

It was seen that the DET teachers regarded the steps of the PSS model in compliance with preparation and implementation of lesson plans. The recommendations given in Table 7 target the lesson plans to be prepared according to the PSS model. The recommendations particularly imply the need to break the problem status in the activities into smaller pieces for progress from the simplest to the most complicated and to allocate as much time as possible to students for their own learning.

Discussion and Conclusion

In this study, first, the steps that can be used in programming teaching were determined by the DT through document analysis. After that, the lesson plans drafted for assessment of these steps were reviewed by the DET and their opinions were collected. Based on the document analysis by the DT and the views of the DET, the steps of the PSS model were established for improving algorithmic thinking, problem solving, and programming skills of learners at secondary school level. The steps of the model were prepared considering both individual and group work as required.

The steps of the model are:

1. Understand the problem
2. Devise a plan
3. Compare the strategies
4. Devise an algorithm
5. Code the algorithm
6. Identify and correct the error in a different code
7. Prepare and code new algorithms

Understand the Problem: This stage requires students to understand what is given in the problem and what they want by determining their place in the solution process. Therefore, students need to improve their abstraction and analogy skills. Abstraction means discarding of unnecessary and insignificant information from the problem by the student. Analogy (Simulation) refers to matching of an unknown concept with a concept known to students for teaching of the former. Abstraction must be performed by students, while analogy by teachers. This phase is performed in a classroom without a computer.

Devise a Plan: It is the stage of determining the steps necessary to solve the problem. At this stage, students try to determine the appropriate way to solve the problem by thinking and trying different strategies. It is carried out as a continuation of the previous step in the class environment and without a computer.

Compare the Strategies: Students compare the strategies they determined in the previous step with the solutions and suitability of the strategies developed by their classmates. The strategy comparison phase covers comparing different solution strategies to find out why the steps are used, what role they play in the solution process, and how they relate to each other. Students should be able to understand not only their own strategies, but also the strategies of other students. In this step, the students understand the necessity of the steps taken for the solution, the relationship with the next steps and the results. The order of strategy steps for this solution is settled. It is carried out as a continuation of the previous step in the class environment and without a computer. Detailed information can be accessed from Şahin (2018).

Devise an Algorithm: This stage consists of 2 sub-steps.

i. Writing the algorithm: Students are instructed to write step by step the solution strategy they determined through comparison in the previous step. Students' use of their own statements for the solution can help them understand the solution.

ii. Playing the algorithm (Drama): A problem in the analogous structure can be revived with drama as it will be related to the daily life of the students. This may trigger inclusion of students with different types of intelligence and increase students' motivation. Therefore, drama activity is considered important for students to express themselves.

The abovementioned steps are carried out without computers in the classroom as continuation of the previous stage. The following stages include computerized processes.

Code the Algorithm: At this stage, students need to encode the algorithm they create by using a computer.

Coding is a task that is perceived difficult by students and requires high-level thinking skills (Jenkins, 2002; Kinnunen & Malmi, 2008). It is a must to comprehend the relationship of each line of code used in algorithm coding to the code in the previous and next line, and the task of that code within the whole program. In other words, each line of code used should be evaluated in terms of the reason for use and the result it will generate for the whole program (Bayman & Mayer, 1988).

Identify and Correct the Error in a Different Code: Understanding and correcting the problem in a different encoding requires an upper level skill than coding our own algorithm. So it is necessary to understand the way in which the person writing the code thinks. As a different activity, it is performed at the computer.

Prepare and Code New Algorithms: At this stage, the students are asked to encode a problem status which they will specify, with the minimum commands they need to write a program only. In this way, it is aimed that the students will be able to learn about the previous processes by doing themselves and being aware of their own progress. The minimum commands are set to prevent students from designing too easy activities. At this stage, if the teacher considers appropriate for the students' level, might ask them to solve problems which are expressed in general terms without any predetermined limitations. The teacher can assign the activity at this stage as an extracurricular activity.

In the abovementioned steps, one of the most important things is to provide students with problems compliant with analogies. It is also believed that if they devise their own strategies and both note down and discuss their strategies in the classroom, it will improve students' ability to express themselves. It can also trigger development of different cognitive processes. In this regards, the relationship between the basic skills and PSS steps in this study is as shown in Figure 4.

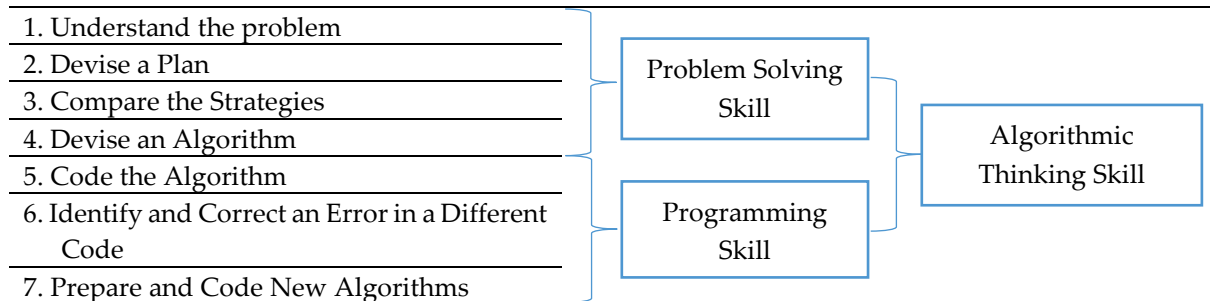


Figure 4. PSS Steps and Associated Cognitive Skills

As shown in Figure 4, the first four steps of the 7-step programming model cover the steps to improve problem solving skills, while the other three steps are aimed at the development of programming skills, which deals with all of the three crucial skills. The comparison of the PSS model steps with the three cognitive skills is shown in Table 8.

Table 8. PSS Relation to Algorithmic Thinking, Problem Solving, and Programming Levels

PSS Method	Sub-dimensions of algorithmic thinking ability (Futschek, 2006)	Steps of Problem Solving Ability(George Polya, 1957)	Programming Levels
1. Understand the problem	Ability to analyze the given problem	1. Understand the problem	Beginner
	Ability to fully comprehend and express the problem (abstraction)		
2. Devise a Plan	Generating basic strategies appropriate for the given problem	2. Devise a Plan	
		3. Carry out the Plan	
3. Compare the Strategies	Ability to think of all possible normal and special cases of a problem	4. Look Back	
4. Devise an Algorithm	Ability to create a correct algorithm using the strategies specific for a problem		
5. Code the Algorithm			
6. Identify and Correct the Error in a Different Code	Ability to increase accuracy of an algorithm		Intermediate
7. Prepare and Code New Algorithms			Advanced

Table 8 shows that the first step of the PSS model, is associated with the lower level sub-dimensions of algorithmic thinking skills as "Ability to analyze the given problem" and "Ability to fully comprehend and express the problem (abstraction)". It is also in connection with Polya's first step of problem solving as "Understand the problem". At those stages, the content of the problem is analysed, the given and wanted data are asked, and unnecessary data are omitted in order to solve any given problem, expressing the obtained data in this way.

The second step of our model corresponds to the sub-dimension of algorithmic thinking as "Generating basic strategies appropriate for the given problem" and Polya's "Devise a Plan" and "Carry out the Plan". These steps cover developing a strategy for solution of a problem and implementing the strategy.

The third step of the model is correlative to algorithmic thinking in sub-dimension "Ability to think of all possible normal and special cases of a problem" and Polya's problem-solving step "Look Back". At the relevant steps, the strategies developed and results of application of the strategies are discussed and evaluated. Decision is made for the best strategy to solve the problem.

The next step in the PSS model relates to algorithmic thinking in the sub-dimension "Ability to create a correct algorithm using the strategies specific for a problem". This step is typical for writing the algorithm of the problem solution by following the most appropriate strategy.

The fifth and sixth steps in our model relate to the algorithmic thinking sub-skill "Ability to increase accuracy of an algorithm". At these steps, improvements can be made on the algorithm for a more efficient outcome through application. Step five in the PSS model addresses to beginner learners of programming. Students at this level typically build their solutions on this structure by adhering to the algorithm devised for solving the problem (Weiser & Shertz, 1983). Likewise, in the fifth step of the SPP method, coding is performed in adherence to the algorithm created at step four.

The sixth step of our model appeals to programming students at intermediate level. In the literature, the 5 operations performed during programming are listed as;

- 1- Identifying problem requirements, developing the solution in the most appropriate way for the problem, and configuring the mental design of these operations
- 2- Being able to understand an algorithm or program previously written by others
- 3- Performing coding by using a computer for solution of the problem
- 4- Being able to find out the reason for a functional dysfunction of a program (Identifying errors)
- 5- Being able to make necessary amendments on a certain program in accordance with needs (Refinement) (Shneiderman, 1976; Koubek, Salvendy, Dunsmore, & Lebold, 1989).

Of the common operations in the programming process, the first one refers to be able to produce the most appropriate solution for a problem and the second is to understand a written program. In order to be able to detect and correct errors in a written program, it is necessary to understand the program. The third operation is coding for solving the problem. The fourth and fifth steps concern identifying the errors on the program and making necessary updates.

Feddon and Charness (1999) investigated the relationship between individuals at different levels of programming and operations performed in programming. They discovered a meaningful relationship between intermediate level programmers and skills of identifying errors (operation four) and editing (operation five) in their study. Bearing this in mind, the study argues that students should be taught to identify and correct errors as one of the programming operations in order to improve programming skills. Similarly, Masuck, Alves-Foss, and Oman (2008) stated that students who have the ability to write programs but are not capable of correcting the errors in the program stay behind at a critical stage of programming.

The last step of the PSS model appeals to students at advanced programming level because students at this level are able to prepare and code the algorithms they need for a problem status on a computer. Weiser and Shertz (1983) noted that advanced programmers are able to write and then code the algorithm which is needed to solve a problem.

In the lesson plans prepared in accordance with the PSS model, tasks expected from teachers and students were specified and explicated with necessary directives. In addition, the forms and examples that teachers can use were attached (Şahin, 2018). The activities in the lesson plan were designed as a guide for teachers in a way allowing adjusting of the difficulty level. As a result, teachers are free to update the activities according to students' cognitive levels, duration, and physical possibilities of their classroom. In this study, an overview of the lesson plans presented to the DET teachers and their use are given in Annex-3.

In conclusion, this study gives an account of generation and steps of the model called Programming in Seven Steps (PSS) and teachers' views on the model as a tool for developing algorithmic thinking, problem solving, and programming skills of students at schools. The model suggested here is expected to yield positive results in in-class implementations.

Recommendations

- The lesson plans based on the PSS model should be applied to students from various success levels to evaluate the effectiveness of the model.
- The lesson plans based on the PSS model should be applied at various grades of education to evaluate the effectiveness of the model.
- The impact of the lesson plans based on the PSS model on algorithmic thinking, problem solving, and programming skills of students should be investigated.

References

- Akpınar, Y., & Altun, A. (2014). Bilgi toplumu okullarında programlama eğitimi gereksinimi. *İlköğretim Online*, 13(1), 1-4.
- Ala-Mutka, K. (2004). *Problems in learning and teaching programming*. Retrieved form https://www.cs.tut.fi/~edge/literature_study.pdf
- Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75, 661-670.
- Balanskat, A., & Engelhardt, K. (2015). *Computing our future: Computer programming and coding - Priorities, school curricula and initiatives across europe*. Retrieved form http://fcl.eun.org/documents/10180/14689/Computing+our+future_final.pdf/746e36b1-e1a6-4bf1-8105-ea27c0d2bbe0
- Bayman, P., & Mayer, R. E. (1988). Using conceptual models to teach BASIC computer programming. *Journal of Educational Psychology*, 80(3), 291-298.
- Bergersen, G. R., & Gustafsson, J. E. (2011). Programming skill, knowledge, and working memory among professional software developers from an investment theory perspective. *Journal of Individual Differences*, 32(4), 201-209.
- Brown, Q., Mongan, W., Kusic, D., Garbarine, E., Fromm, E., & Fontecchio, A. (2013). *Computer aided instruction as a vehicle for problem solving: Scratch programming environment in the middle years classroom*. Retrieved form http://www.pages.drexel.edu/~dmk25/ASEE_08.pdf
- Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. *ACM SIGCSE Bulletin*, 33(3), 49-52.
- Choi, J. Lee, Y., & Lee, E. (2016). Puzzle based algorithm learning for cultivating computational thinking. *Wireless Personal Communications*, 93(1), 131-145. doi: 10.1007/s11277-016-3679-9
- Clements, D., & Sarama, J. (2003). Strip mining for gold: research and policy in educational technology a response to "fool's gold". *AACE Journal*, 11(1), 7-69. Retrieved form https://www.learntechlib.org/index.cfm/files/paper_17793.pdf?fuseaction=Reader.DownloadFullText&paper_id=17793%C2%A0
- Committee on Logic Education. (2008). *Algorithmic thinking*. Retrieved form <http://www.ucalgary.ca/aslcle/nctm/Q2A.html>
- Crescenzi, P., Malizia, A., Verri, M. C., Diaz, P., & Aedo, I. (2012). Integrating algorithm visualization video into a first-year algorithm and data structure course. *Educational Technology & Society*, 15(2), 115-124.
- Creswell, J. W. (2013). Steps in conducting a scholarly mixed methods study. *DBER Speaker Series*, 48. Retrieved form <http://digitalcommons.unl.edu/dberspeakers/48>
- Coull, N. J., & Duncan, I. M. (2011). Emergent requirements for supporting introductory programming. *Innovation in Teaching and Learning in Information and Computer Sciences*, 10(1), 78-85.
- Çıray, F., & Erişti, B. (2014). Disiplinlerarası analogi tabanlı öğretimin farklı düzeylerde akademik başarılı ilköğretim öğrencilerinin fen ve teknoloji dersi öğrenme düzeyleri üzerindeki etkisi. *İlköğretim Online*, 13(3), 1049-1064.
- Fesakis, G., & Serafeim, K. (2009). Influence of the familiarization with scratch on future teachers' opinions and attitudes about programming and ICT in education. *ACM SIGCSE Bulletin*, 41(3), 258-262.

- Feddon, J. S., & Charness, N. (1999). Component relationships depend on skill in programming. *11th Annual PPIG Workshop, University of Leeds, UK*, 1-11. Retrieved from <https://pdfs.semanticscholar.org/dccf/2f3cd095192abd0b5c624c5dda7948f5826b.pdf>
- Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. In R. T. Mittermeir (Ed.), *Informatics Education – The Bridge between Using and Understanding Computers* (V. 4226, pp. 159-168). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/11915355_15
- Garner, S. (2003). Learning resources and tools to aid novices learn programming. In *Informing science & information technology education joint conference (INSITE)* (pp. 213-222). Retrieved from <https://pdfs.semanticscholar.org/21a6/68fb94878b040e4bffba0858d15896cbbdb8.pdf>
- Ginat, D. (2004). On novice loop boundaries and range conceptions. *Computer Science Education*, 14(3), 165-181.
- Grover, S., & Pea, R. D. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43.
- Helminen, J., & Malmi, L. (2010). Jype - a program visualization and programming exercise tool for Python. In *Proceedings of the 5th international symposium on Software visualization - SOFTVIS '10* (pp. 153). Salt Lake City, Utah, USA: ACM Press. <https://doi.org/10.1145/1879211.1879234>
- ISTE. (2016). *The ISTE National Educational Technology Standards (NETS) and Performance Indicators for Students*. Retrieved from <http://www.iste.org/standards/nets-for-students>
- Jenkins, T. (2002, September). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*. Leeds.
- Kam, H. J., & Katerattanakul, P. (2014). Structural model of team-based learning using Web 2.0 collaborative software. *Computers & Education*, 76, 1-12.
- Karaman, S., Yıldırım, S., & Kaban, A. (2008). Öğrenme 2.0 yaygınlaşıyor: Web 2.0 uygulamalarının eğitimde kullanımına ilişkin araştırmalar ve sonuçları. In *XIII. Türkiye’de İnternet Konferansı Bildirileri* (pp. 35-40). Orta Doğu Teknik Üniversitesi, Ankara.
- Kalelioglu, 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
- Kafai, Y. B., & Q. Burke. (2014). *Connected code: Why children need to learn programming*. MIT Press.
- Kinnunen, P., & Malmi, L. (2008, September). *CS minors in a CS1 course*. Paper presented at the Fourth international Workshop on Computing Education Research, Sydney.
- Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM*, 1455-1464.
- Klopfer, E., & Yoon, S. (2005). Developing games and simulations for today and tomorrow's tech savvy youth. *TechTrends*, 49(3), 33-41.
- Kobsiripat, W. (2015). Effects of the media to promote the scratch programming capabilities creativity of elementary school students. *Procedia-Social and Behavioral Sciences*, 174, 227- 232.
- Koubek, R. J., Salvendy, G., Dunsmore, H. E., & LeBold, W. K. (1989). Cognitive issues in the process of software development: review and reappraisal. *International Journal of Man-Machine Studies*, 30, 171-191.

- Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1989). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research*, 2(4), 429-458.
- Kukul, V., & Gökçearslan, Ş. (2014). *Scratch ile programlama eğitimi alan öğrencilerin problem çözme becerilerinin incelenmesi*. 8. Uluslararası Bilgisayar ve Öğretim Teknolojileri Sempozyumu'nda sunulan bildiri, Trakya Üniversitesi Bilgisayar Öğretmenliği Bölümü, Edirne.
- Law, K. M., Lee, V. C., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, 55(1), 218-228.
- Lahtinen, E., Ala-Mutka, K., & Jarvinen, H. (2005) A Study of Difficulties of Novice Programmers. In *Acm Sigcse Bulletin*, ACM, 37(3), 14-18.
- Lai, A. F., & Yang, S. M. (2011). The learning effect of visualized programming learning on 6th graders' problem solving and logical reasoning abilities. In *2011 International Conference on Electrical and Control Engineering* (pp. 6940-6944). Yichang, China: IEEE. <https://doi.org/10.1109/ICECENG.2011.6056908>
- Lai, C. S., & Lai, M. H. (2012). Using computer programming to enhance science learning for 5th graders in taipei. In *2012 International Symposium on Computer, Consumer and Control* (pp. 146-148). Taichung, Taiwan: IEEE. <https://doi.org/10.1109/IS3C.2012.45>
- Lee, Y. J. (2011). Empowering teachers to create educational software: A constructivist approach utilizing Etoys, pair programming and cognitive apprenticeship. *Computers & Education*, 56(2), 527-538.
- Linn, M. C., & Dalbey, J. (1989). Cognitive consequences of programming instruction. In E. Soloway & J.C. Spohrer (Eds.), *Studying the novice programmer* (pp. 57-81). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, 39(1), 223-227.
- Masuck, C., Alves-Foss, J., & Oman, P. (2008). Analysis of fault models for student use. *ACM SIGCSE Bulletin*, 40(2), 79-83.
- Merriam, S. B., & Tisdell, E. J. (2015). *Qualitative research: A guide to design and implementation*. John Wiley & Sons.
- Milli Eğitim Bakanlığı. (2017). *Bilişim teknolojileri ve yazılım dergisi öğretim programı, 2016-2017*. Ankara: Milli Eğitim Basımevi.
- Naharro-Berrocal, F., Pareja-Flores, C., Urquiza-Fuentes, J., & Velázquez-Iturbide, J. á. (2002). Approaches to comprehension-preserving graphical reduction of program visualizations. In *Proceedings of the 2002 ACM symposium on Applied computing-SAC '02* (pp. 771). Madrid, Spain: ACM Press. <https://doi.org/10.1145/508791.508941>
- Oluk, A., & Saltan, F. (2015). Effects of using the scratch program in 6th grade information technologies courses on algorithm development and problem solving skills [Special issue]. *Participatory Educational Research*, 10-20.
- Özmen, B., & Altun, A. (2014). "Undergraduate students' experiences in programming: difficulties and obstacles." *Turkish Online Journal of Qualitative Inquiry*, 5(3), 1-27.
- Porter, R., & Calder, P. (2004). Patterns in learning to program: an experiment?. In *Proceedings of the Sixth Australasian Conference on Computing Education - Volume 30* (pp. 241-246). Darlinghurst, Australia, Australia: Australian Computer Society, Inc.
- Polya, G. (1957). *How to solve It?* (2nd ed.). Princeton, N.J.: Princeton University Press.

- Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137-172.
- Sanford, J. F., & Naidu, J. T. (2016). Computational thinking concepts for grade school. *Contemporary Issues in Education Research (Online), 9*(1), 23.
- 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.", *Computer & Education, 97*, 129-141.
- Scaffidi, C., & Chambers, C. (2012). Skill progression demonstrated by users in the Scratch animation environment. *International Journal of Human-Computer Interaction, 28*(6), 383-398.
- Schwartz, J., Stagner, J., & Morrison, W. (2006). Kid's programming language (Kpl). In *ACM SIGGRAPH 2006 Educators program on-SIGGRAPH '06* (pp. 52). Boston, Massachusetts: ACM Press. <https://doi.org/10.1145/1179295.1179348>.
- Seppälä, O., Malmi, L., & Korhonen, A. (2006). Observations on student misconceptions—A case study of the Build-Heap Algorithm. *Computer Science Education, 16*(3), 241-255.
- Shneiderman, B. (1976). Exploratory experiments in programmer behavior. *International Journal of Computer and Information Sciences, 5*, 123-143.
- Shih, I. J. (2014). *The effect of scratch programming on the seventh graders' mathematics abilities and problem solving attitudes* (Yayımlanmamış yüksek lisans tezi). Taipei University, Taiwan.
- Silverman, D. (2013). *Doing qualitative research: A practical handbook*. London: SAGE Publications.
- Spohrer, J. C., & Soloway, E. (1989). Simulating student programmers. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1* (pp. 543-549). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Stolee, K. T., & Fristoe, T. (2011). Expressing computer science concepts through Kodu game lab. In *Proceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE '11* (pp. 99). Dallas, TX, USA: ACM Press. <https://doi.org/10.1145/1953163.1953197>
- Şahin, G. (2018). *Ortaokul seviyesinde programlama öğretimi için bir yöntem önerisi* (Yayımlanmamış yüksek lisans tezi). Karadeniz Teknik Üniversitesi, Eğitim Bilimleri Enstitüsü, Trabzon.
- Trilling, B., & Fadel, C. (2012). *21st century skills: learning for life in our times* (1st ed.). San Francisco, Calif: Jossey-Bass.
- Utting, I., Cooper, S., Kölling, M., Maloney, J., & Resnick, M. (2010). Alice, greenfoot, and scratch-a discussion. *ACM Transactions on Computing Education, 10*(4), 1-11.
- Vasconcelos, J. (2007). *Basic Strategy for Algorithmic Problem Solving*. Retrieved from <http://www.cs.jhu.edu/~jorgev/cs106/ProblemSolving.htm>
- Van Gorp, M. J., & Grissom, S. (2001). An empirical evaluation of using constructive classroom activities to teach introductory programming. *Computer Science Education, 11*(3), 247-260.
- Weiser, M., & Shertz, J. (1983). Programming problem representation in novice and expert programmers. *International Journal of Man-Machine Studies, 19*(4), 391-398.
- Winslow, L. E. (1996). Programming pedagogy-a psychological overview. *ACM Sigcse Bulletin, 28*(3), 17-22.
- Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education, 12*(3), 197-212.

- Wilson, A., & Moffat, D.C. (2010). *Evaluating Scratch to introduce younger school children to programming*. Retrieved from <http://scratched.gse.harvard.edu/sites/default/files/wilson-moffat-ppig2010-final.pdf>
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.
- Wu, W. Y., Chang, C. K., & He, Y. Y. (2010). Using Scratch as game-based learning tool to reduce learning anxiety in programming course. Global Learn sunulan sözlü bildiri, Association for the Advancement of Computing in Education (AACE).
- Yuretich, R. F., Khan, S. A., Leckie, R. M., & Clement, J. J. (2001). Active-learning methods to improve student performance and scientific interest in a large introductory oceanography course. *Journal of Geoscience Education*, 49(2), 111-119.
- Yükseltürk, E., Altok, S., & Üçgül, M. (2016). *Oyun programlamanın ilköğretim öğrencilerinin problem çözme becerilerine etkileri: Bir yaz kampı deneyimleri*. 4. Uluslararası Öğretim Teknolojileri ve Öğretmen Eğitimi Sempozyumu'nda sunulan sözlü bildiri, Fırat Üniversitesi Bilgisayar ve Öğretim Teknolojileri Bölümü, Elazığ.
- Yıldırım, A., & Şimşek, H. (2011). *Sosyal bilimlerde nitel araştırma yöntemleri*. İstanbul: Seçkin Yayınları.
- Yin, R. K. (2017). *Case study research and applications: Design and methods*. Los Angeles: SAGE.
- Zsakó, L., & Szlávi, P. (2012). ICT competences: algorithmic thinking. *Acta Didactica Napocensia*, 5(2), 49-58.
- Ziatdinov, R., & Musa, S. (2012). Rapid mental computation system as a tool for algorithmic thinking of elementary school students development. *European researcher, Series A*, (7), 1105-1110.

Appendix 1. Lesson Plan Evaluation Criteria Form

Görev yaptığınız okulun adı:
Branşınız:
Öğretmenlik deneyiminiz:

1. Ders planı hazırlarken dikkat edilmesi gereken noktalar nelerdir?

Appendix 2. Interview Questions Used by the IT and Software Teachers

1. How compliant is the activity in the lesson plan for teaching of programming to secondary 6th graders with the students' levels?
2. How compliant are the computer applications in the lesson plan for teaching of programming to secondary 6th graders with the students' levels?
3. Is the duration allocated for the activities in the lesson plan for teaching of programming to secondary 6th graders sufficient? Why?
4. Is the duration allocated for the computer applications in the lesson plan for teaching of programming to secondary 6th graders sufficient? Why?
5. How feasible are the activities in the lesson plan for teaching of programming to secondary 6th graders in overcrowded classrooms?
6. How feasible are the computer applications in the lesson plan for teaching of programming to secondary 6th graders in overcrowded classrooms?

Appendix 3. An Overview of Lesson Plans and Use of the Schedule

Table 9 summarizes the lesson plans in dimensions of topics, attainments and activities and provides overall information about implementation of the lesson plants in accordance with the 7 steps.

Table 9. Summary Table for Lesson Plans

Topic	Attainment	Activity	Step	Type	Individual or Group	Duration	Week
Algorithm Concept	<ul style="list-style-type: none"> Explains the basic concepts of problem solving process. Expresses the importance of correct identification and ordering of the solution stages of a problem. Proposes ways of solution for everyday problems. Designs different algorithms to solve the given problem. Analyzes the algorithm and predicts the results. Eliminates the troubles in the algorithm. Adjusts the algorithm for increased efficiency. Develops ideas with peers during the problem analyzing and solving stages. 	Helping the Shepherd	1-7	Non-computer	Individual or Group	40 mins	1
		Weaver Birds	1-7	Non-computer	Individual or Group	40 mins	
Let's Recognize the Programming Environment	<ul style="list-style-type: none"> Adds a character to the screen. Changes the screen background. Gives movement to the character. Gives movement by changing character costume. 	Leisure Time	1-7	Computer-aided	Individual	160 mins	2-3
Variables	<ul style="list-style-type: none"> Determines the constants and variables in the given problem. Uses the constants and variables to solve the given problem. Describes types of data. Explains the difference between data types. Uses data types to solve the given problem. 	"Buying fruit at greengrocer"	1	Non-computer	Individual or Group	5 mins	4-5
		"I choose fruit"	2-4	Non-computer	Group	35 mins	
		"My fruit basket"	5-7	Non-computer	Individual	120 mins	
Conditionals	<ul style="list-style-type: none"> Explains the decision logic structure. Explains the multiple if structure. 	"Map" Activity	1	Non-computer	Individual or Group	10 mins	6-7


<ul style="list-style-type: none"> • Explains and exemplifies plain logic. • Explains and exemplifies positive logic. • Explains and exemplifies negative logic. • Converts logical structures to each other. • Forms decision tables. • Designs algorithms using decision making in problem solving processes. • Creates flowchart using decision making in problem solving process. 	"Colourful Steps"	2-4	Non-computer	Group	30 mins	
	"Maze map game"	5-7	Computer-aided	Individual	120 mins	
Cycles	<ul style="list-style-type: none"> • Explains the cycle logic structure. • Explains the function of the increment values in the cycle logic structure. • Explains the conditions in the cycle logic structure. • Designs algorithms suitable for cycle logic structure. • Designs algorithms using cycle structures in problem solving process. 	"Ali playing a game"	1	Non-computer	Individual or Group	5 mins
	"Auto racing" in-class animation	2-4	Non-computer	Group	35 mins	
	"My fuel amount"	5-6	Computer-aided	Individual	115 mins	
	"Auto racing" app	7	Computer-aided	Individual	5 mins	

Appendix 4. Regarding cycles, the PAT and SOT forms are as follows.

- Form for Understand the Problem

#	Questions	Answers
1.	What types of tracks are in the game?	
2.	What are the colors of vehicles in the game?	
3.	How many rounds does the game have?	
4.	What is the purpose of the game?	

- Form for Devising a Plan

Road	Color of the vehicle	Time en route	Exchange of vehicle	Duration of vehicle exchanging
Earth				
Asphalt				
Sand				
Pebble				
				
One-way duration:			Exchanging duration:	
Round-way duration:			Total duration:	