Informatics in Education, 2020, Vol. 19, No. 2, 181–200

DOI: 10.15388/infedu.2020.09

Effect of Using Metacognitive Strategies to Enhance Programming Performances

Ünal ÇAKIROĞLU, Betul ER

Trabzon University, Computer and Instructional Technologies Department
Sogutlu, 61300, Trabzon, Turkey
e-mail: cakiroglu@trabzon.edu.tr, betulyildirim05@gmail.com

Received: June 2019

Abstract. Considerable effort has been invested in innovative practices about teaching programming. Although the usefulness of metacognition in learning process is acknowledged, evidence demonstrating how metacognitive strategies effect in the programming classrooms is still very scarce. Given the importance of metacognitive strategies, this study seeks to examine the effect of the strategies to students' performances in programming courses. The qualitative techniques were used to determine the participants' programming performances and explicate their experiences about the role of the strategies. The results indicated that while almost half of the students' programming performances were multistructural the other half was prestructural and unistructural categories of Solo taxonomy. The quality of the programming problems is found to have an important role in the development of both cognitive knowledge and cognitive regulation strategies. Furthermore, the cognitive potentials and problem solving habits of the students were also found to be effective on their metacognitive development. The implications of notable findings and directions for future studies were also discussed.

Keywords: metacognitive strategies, programming performances, problem solving.

Introduction

Scholars have been using innovative instructional techniques in the field of teaching programming. In the last decades, programming has been taught as a mandatory course in many computer science departments. Researchers claim that learning programming for the first time is challenging (Ford, 2007, Helminen and Malmi, 2010; Chen, Li and Wang, 2012). While programming, students should have basic programming language knowledge and problem-solving skills (Gundurao, Manjunath and Nachappa, 2010). During the programming process, it is difficult for students to manage their conceptual, operational and procedural knowledge in programming environment. In this context, in order to facilitate learning process, teaching methods have gained much attention in recent years. For the last three decades, new perspectives in teaching and assessing programming have gained popularity among educators.

During the programming processes, students should think about the solutions analytically, perform proper programming approaches and correct the programming errors. While seeking alternative solutions, they should decide about the actions in order. In order to acquire acquisitions, students should examine the program outputs thoroughly and transfer their gains to other problems (Guss and Wiley, 2007; Linn and Dalbey, 2013; Mohd Rum, Ismail, 2014). Since learning or doing programming is a challenging process, students should take responsibility of their own learning and enable their metacognition and self-regulation. Many researchers argue that enabling metacognition in the problem-solving positively influences the process. For instance; the study conducted by Teong (2003) noted the significant effects of meta-cognitive training on problem solving. Thus, it could be claimed that improving metacognition in programming can facilitate the problem-solving process via programming. In this study, we thought that triggering students' metacognitive skills can positively affect programming performances. In order to achieve this, we enriched the programming tasks with metacognitive strategies to make the problem-solving processes easier. In line with this idea, we designed an intervention to improve students' metacognitive awareness and make them use metacognitive strategies in the programming class.

Metacognition in Programming

Metacognition is defined as an individual's awareness of own cognitive processes and capability of controlling these processes (Flavell, 1979; Livingston, 2003; Scott and Levy, 2013). A student who has metacognitive skills knows how his/her cognitive system works and monitors his/her own performance (i.e. self-monitoring). Researchers argue that one of the most important factors leading learning is activating metacognition (Schraw, 1998; Veenman, Prins, and Elshout, 2002). Metacognition is discussed into two dimensions as cognition and regulation of cognition (Schraw and Moshman, 1995). According to this model, cognition knowledge is defined as the knowledge about one's own cognition. According to Schraw and Moshman (1995), the declarative knowledge strategy is the knowledge of the individual in general about himself, his strategies and the factors that affect his performance. Procedural knowledge strategy is the knowledge of the individual's ability to perform and follow the steps of a procedure. Conditional knowledge strategy is an individual's knowledge of when and how cognitive activities are implemented.

The regulation of cognition consists of strategies that guide the individual's own learning. Planning, monitoring and evaluation are three basic metacognitive strategies about cognition regulation. Planning refers to the selection of targeted strategies and effective selection of the resources necessary to achieve the target (Miller, 1985). Monitoring is the strategy that covers the progress of the individual by evaluating whether s/he comprehends his/her goal during the learning process (Schraw, 1998). Evaluation is the decision about the learning outcomes and the effectiveness of the strategies in the learning process (Schraw and Moshman, 1995).

In order to develop metacognitive skills, teachers are suggested to provide metacognitive support to students. Researchers suggest that students should be provided with appropriate learning environments for developing and implementing metacognitive skills (Derry, 1992; Salomon and Perkins, 1989). In this context, combining metacognitive strategies and instruction is crucial for effective learning outcomes (Hartley, 2001; Park, 1992). Hence; enhancing metacognitive awareness is critical and it has received a growing attention. The fact that metacognitive skills can be improved by some approaches leads researchers to study on the relationship between metacognitive strategies and problem-solving skills.

Problem-solving in computer programming requires cognitive skills that the students should work in a methodical manner and provide special representations. Programming process demands metacognitive skills because programmers should organize their knowledge and use the features of programming environment at the same time. Programmer should decide on the approach they would use, the strategies they would apply and they should organize the structural programming knowledge. They should learn from the feedback, error messages or other tools during programming that is suitable for developing metacognition skills and awareness.

According to Storey (2004), during programming, the programmer creates a mental model, expresses the data and the working system of the program, and thinks algorithmic (Apiola and Tedre, 2012; Zapu ek and Rugelj, 2013). Making assumptions and formulizing the modifications (Helminen and Malmi, 2010) and managing the ideas during problem-solving are also parts of programming (McCormick, 2003; Lee, Teo and Bergin, 2009. They). Students often need support to eliminate difficulties in problem solving processes. In this sense, metacognitive controls can facilitate programming process in planning (solution plan), monitoring (monitoring the design and development of the program) and evaluation (testing the problem solution and reflecting it) (Lee, Teo and Bergin, 2009). Within this idea, many alternative programming environments have been designed with the goal of supporting novices while learning to program. A few research in programming found that students who possess metacognitive management skills and strategies perform well in programming compared to those in whom the skills are lacking (Bergin, Reilly, and Traynor, 2005).

While a large number of studies have been conducted to investigate the role of meta-cognition in the learning process, there have been few investigations in the context of learning programming. Following conclusions from the previous studies (Lin, Schwartz and Hatano, 2005), we aim to gain a practical insight into the relationships between training metacognitive strategies and programming performances. Thus; the purpose of this study is to investigate the effect of efforts in developing metacognitive strategies on the programming performances. More specifically, our research addressed the following question:

How do metacognitive strategies affect the programming performances during higher education programming course?

Research Methodology

In this case study, qualitative data collection and analysis procedure is carried out. Multiple data sources are used to determine the role of metacognition in programming performances (Yin, 2003). A small sample group is chosen to meet the certain purpose of the qualitative design. The sample group of this study consists of 16 candidates of IT teachers (7 females, 9 males, average age = 20) enrolled in an IT department of a Turkish university. All of the participants received introductory programming course in the previous year. Participants are being referred to in an alphanumeric manner as S1, S2, ...

Instructional Process

Problem-solving is a common activity used as a teaching and learning approach in computer programming education because it helps students to develop different cognitive abilities. Solving problems requires a great deal of cognitive effort in activities such as finding solutions, identifying problems, and testing hypotheses. The study was conducted with 10 projects presented to the students in Advanced Programming Course which covers basic programming concepts and structures. The projects are presented in Table 1. Learning objectives include the concepts of basic variables and transactions, loop and comparison, inner loop and operators, conditionals (if-else), functions, arrays and logical reasoning problems. The study lasted for10 weeks.

The study procedure is summarized in Fig. 1.

Data Collection Tools

Data collections tools were used to assess the development of metacognitive skills and also the programming performances.

The data collection tools utilized are shown in Table 2.

Table1
Weekly programming projects

Project #	Projects
P1	Arithmetic Operations
P2	Exponential numbers
P3	Random Number Generation
P4	Armstrong Numbers
P5	Pascal Triangle
P6	Arrays
P7	Number Analysis
P8	Hangman Game
P9	Problems
P10	Chess Pieces

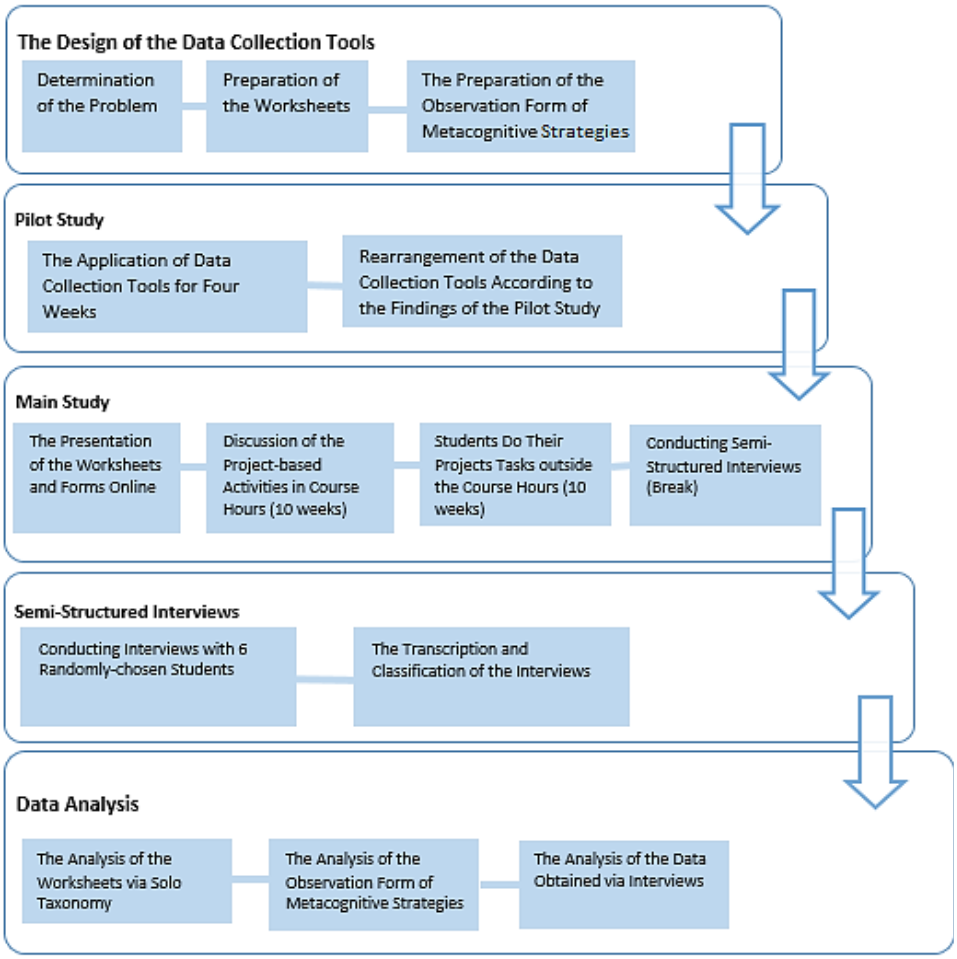


Fig. 1. Research procedure.

Table 2

Data collection tools

Data Collection Tools	Purpose of Use	Frequency of use
Worksheet	Developing metacognitive strategies Evaluating programming performances	Used weekly throughout the process
Observation Form	To observe the development of metacognitive strategies	Used weekly following the worksheets
Interview	To determine the evaluations and opinions of the students for the study and procedure.	During the teaching period and once at the end of the given period

Worksheets

In this study, meta-cognitive training is carried out via worksheets which can be considered as an instructional intervention to develop the practice of meta-cognitive strategies. In the worksheet, metacognitive strategies were presented with metacognitive exercises. The worksheet provides both the enrichment of metacognition and programming teaching and the evaluation of programming skills through solutions. Students' programming performances were evaluated by analysing the responses which they provided for the problems in these worksheets. It was prepared by the researchers through considering the reviews of two experts. A four-week of pilot study with another sample was conducted to rearrange the questions in the worksheets.

In this study, writing a program is considered as a challenging process. Therefore, the metacognitive strategies were delivered at the starting point to write a program, during writing a program and after finishing the writing process. With this in mind, metacognitive support was made regarding these three stages in the worksheet: at the beginning, during and at the end of the program.

The development of students' programming skills was analysed and evaluated within the framework of Solo taxonomy over their answers to the problems in these worksheets. Research shows that SOLO is an effective framework for examining the indicators of the students' programming knowledge and examining their development in programming (Lister *et al.*, 2006; Sheard *et al.*, 2008). Therefore, in recent years, SOLO taxonomy has been widely used in computer science education to classify students' solutions to programming assignments (Whalley *et al.*, 2011).

Observation Form

Metacognitive skills are difficult to measure because there are internal processes that individuals are often unaware of (Schraw, 2009). Measurement tools used to measure metacognition are examined in two categories: reports based on one's own statements (questionnaires and interviews) and objective behaviour measurements (systematic observation and voice thinking protocols) (Veenman, 2005). According to Schraw (2009), it is very difficult to find a method that allows the simultaneous connection to the metacognition process and can measure all of these processes alone. Therefore, the use of such multidimensional scales to measure metacognitive skills is considered appropriate by some researchers (Garner and Alexander, 1989; Schraw, 2009).The observation form in this study was based on the metacognition model suggested by Schraw and Moshman (1995). In this model, metacognition is considered under two main categories; knowledge of cognition and regulation of the cognition. Knowledge of cognition is examined under three sub-categories; declarative knowledge, procedural knowledge and conditional knowledge. The regulation of cognition is evaluated under three sub-categories; planning, monitoring and evaluation. Planning strategy refers to the selection of targeted strategies and effective selection of the resources necessary to achieve the target (Miller, 1985). Monitoring strategy is the strategy that covers progress by evaluating whether the

target is understood during the learning process, how the performance is, and the effectiveness of the strategies used (Schraw, 1998). The evaluation strategy is that the learner makes decisions about the learning outcomes and the effectiveness of the strategies used throughout the learning process (Schraw and Moshman, 1995).

The observation form was used to determine the development of metacognitive skills used during the instructional process. It was developed by the researcher according to the reviews of the field experts. While preparing the observation form, a two-phased test development technique was used. Two-phased tests are developed to eliminate the disadvantages of the multiple-choice tests (Tan, Goh, Chia and Treagust, 2002; Mann and Treagust, 1998). At the first phase, students were asked to determine the case, and at the second phase they were asked to explain why they determined the given case.

The form is 3-point Likert type (Yes, No, Partially) and also includes an open ended item with each item. Before starting to write programs, students' were asked to write their goals, strategies and the resources they used in the worksheet. Thus, students' answers in the worksheet were easily analysed.

Interviews

Interviews were conducted in order to explain the students' responses in the observation form. In line with the reviews of the experts, six open-ended questions were provided in the interviews. The experts were instructors having at least 10 years of programming teaching experience. The interviews enable the researcher to examine students' thoughts in depth. Taking students' answers in the worksheets, students were asked to explain whether metacognition strategies were useful for their development of metacognitive strategies. The interviews were conducted with randomly selected six students once in three-weeks during the research process and with six students at the end of the process.

Data Analysis Methods

Analysis of the Worksheets

In this study; the SOLO Assessment Taxonomy as developed by Lister *et al.* (2006) was adopted and used to evaluate programming performances. Worksheets were applied weekly for the period of ten-weeks with the aim of developing metacognitive strategies of the students. The answers to the worksheet questions below about the tasks were evaluated within the framework of SOLO categories.

1. Review what was asked from you in the question. What is the necessary information you need to have to write code?
2. Which procedures do you perform gradually while writing code?
3. How did this code contribute to your programming knowledge? Where could you use what you learned?

Table3

SOLO assessment taxonomy

SOLO Category	Assessment Criteria
Prestructural	• The answer is either irrelevant or slightly relevant with the question. • Lack of information in terms of programming structures
Unistructural	• Answer explains only a part of the code • The student reflects the correct comprehension but not all aspects of the question. • Code repetition is made from the other programming duties or education materials.
Multistructural	• Answer explains most of the code with minor mistakes. • Answer explains most of the code without focusing on the relationship between codes and expressions.
Relational	• Answer shows that the purpose and function of the code is understood. • Answers show the programming concepts and the integration of the structures. • Understanding how to apply key programming concept/idea to the similar/known problem.
Extended Abstract	• Achieving key program principals or assigned problem and questioning. • Answer explains a program concept or principals in a way which shows problem solving skills of the students on the new or unseen problems. • Obtained information could be integrated into more sophisticated programs/other fields.

The assessment criteria for the analysis of the worksheets and SOLO categories are summed up in Table 3.

The answers of the three questions were combined and evaluated as one since they correspond to one category in SOLO taxonomy. The responses of the students were evaluated according to which criteria they meet in the SOLO. For instance, if the answer of the students was irrelevant or lacks the basic structures of programming, the programming performance of the student was evaluated as prestructural for the given task.

Analysis of the Observation Form

Observation forms were analysed with a rubric prepared according to the model of Schraw and Moshman (1995). In the form, 14 items were included with a 3-point Likert type scale including Y*es, No, Partly* and there was an explanation section to be completed for each item. While assessing the forms, scale and explanation part were evaluated together and item score was determined. The item score scale is presented in Table 4.

The form contains a 3-point Likert scale, Yes, No and Partially, and a description section for each item. The responses on the form were evaluated by regarding the explanations and the responses that the students provided for each item. The responses and explanations were used to discuss the development of using metacognition strategies for students.

When evaluating the forms, the responses to the items and explanation part were evaluated together. When evaluating the item about the status of declarative information from metacognition strategies, the situation of predicting whether the student can perform the given task was taken into consideration. When evaluating the item about

Table 4
Item score scale

Explanation Level	Evaluation	Score
Needs Improvement	Yes	1
	Partly	0
	No	0
Acceptable	Yes	3
	Partly	2
	No	1
Advanced	Yes	4
	Partly	3
	No	2

procedural information, the ability of the student to determine the steps to be used in the program writing process was taken into consideration.

While evaluating the planning strategy, the situation of determining the student's purpose and resources and associating them in a significant way was examined. While evaluating the monitoring strategy in the form, the student scored according to the status of realizing the objectives during the program writing process and determining the necessary steps in order to reach the results in accordance with the project.

In the form, the items were evaluated as advanced if the students' statements clearly indicated that they were able to determine the success of the course, the points they had difficulty, their strengths and weaknesses and the ability to integrate the information they learned in other fields. If the statements partially reflected these situations, the items were considered acceptable. Regarding the criteria in Table 4, the responses in the form were evaluated. For instance, while evaluating the item 2 which allows determining the

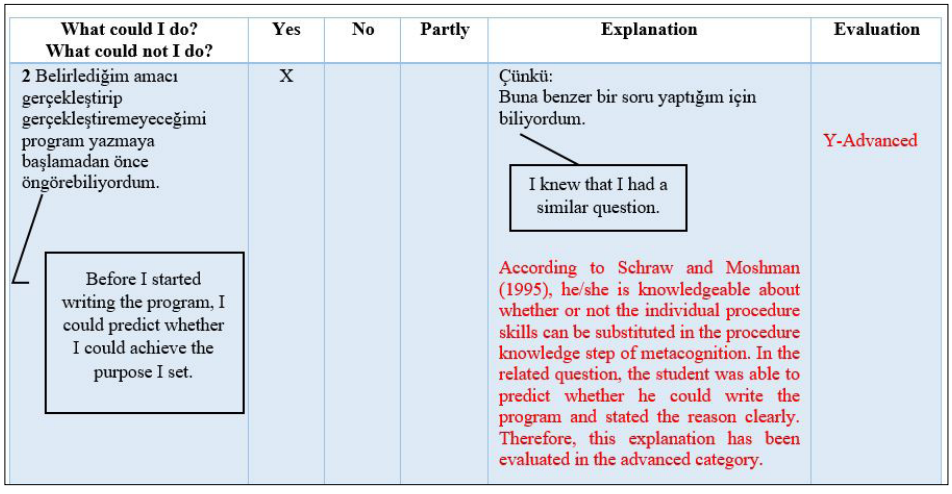


Fig. 2. A sample item from the observation form.

Table 5
Evaluating metacognitive strategy development

	Acceptable	Needs Improvement	Advanced
Declarative Knowledge	Ability to predict achievement of the task and to explain the reason for it.		
Procedural Knowledge	Ability to determine the steps to take while writing a program, to express these steps respectively.		
Conditional Knowledge	Being aware of the necessity of getting help about the given strategies, and to be able to express what this help exactly is.		
Planning	Be able to determine the aims and sources while doing the tasks, and associating them clearly.		
Monitoring	Be able to reveal the progress of doing tasks throughout the process of writing a program, and to determine the necessary steps properly to reach a conclusion.		
Evaluation	Be able to determine course success, the challenging points, strength and weaknesses, and whether s/he can integrate the newly learned information into other fields throughout the program.		

declarative knowledge among the metacognition strategies, it is regarded whether the students performed the given task or not. The evaluation items of metacognitive strategy development is presented in Table 5.

If the student could predict whether s/he could perform the task and could explain the reason for it, the status of the student about the given knowledge was evaluated as "Advanced". On the other hand, if the student could predict whether s/he could perform the task and could not explain the reason for it, the item was evaluated as "Acceptable". On the contrary, if a student indicated not to predict whether s/he could do the task, the item was evaluated as "Needs Improvement". For instance, since the student could not express his/her purpose clearly with the sentence "My purpose: Running the program", it was evaluated as "Acceptable". If the student could not determine these cases, his/her answer was evaluated as "Needs Improvement".

Results

Changes in Metacognitive Skills

Observation form for the development of metacognition strategies was analysed through a rubric. Students' development in two main dimensions which are knowledge of cognition and regulation of cognition are presented in Fig. 3.

Fig. 3 indicates that the scores for knowledge of cognition is at average level (between 5 and 10) during the ten-week. In terms of regulation of the knowledge, there were slight decreases in the second and eighth weeks. S5 explained the reason for this decrease in semi-structured interviews as "Since the solutions for the last projects were challenging, it was helpful in terms of finding new solutions to different questions…".

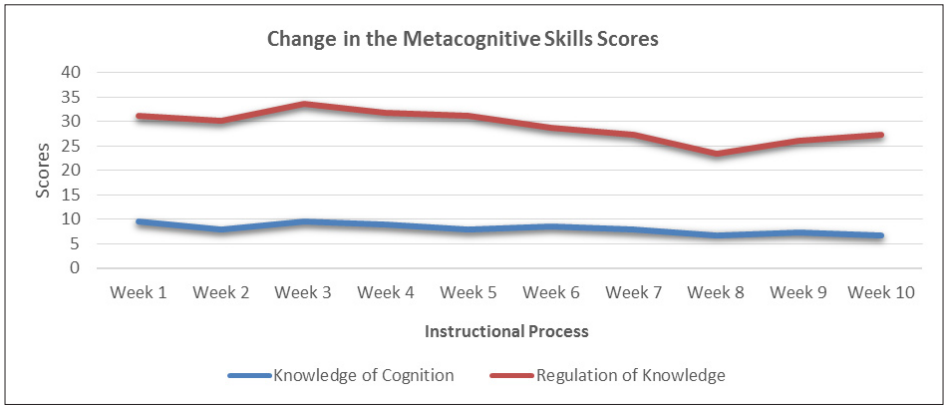


Fig. 3. The development knowledge of the cognition, regulation of the knowledge.

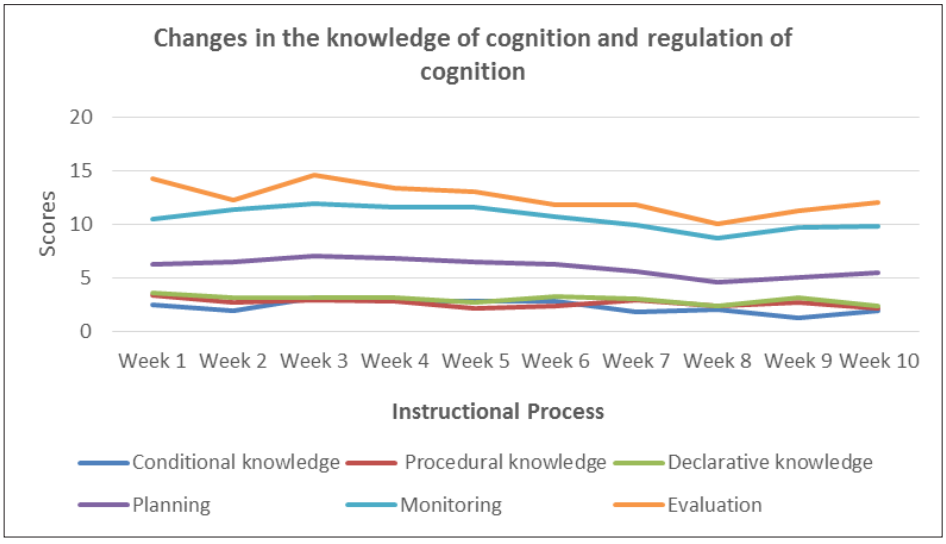


Fig. 4. Changes in the scores of metacognition sub-skills.

The weekly average scores of the 16 students' metacognitive development in these sub-categories are shown in Fig. 4.

Fig. 4 indicates that the average score of declarative knowledge is in the ranks of 2.7 and 3.7 (Acceptable- Advanced). Some of the students who progressed in using the strategy of declarative knowledge indicated that: S14: "I was aware of what the program asked from me and to what extent I could achieve them", S15: "I understood the logic of it and I know I could handle it". Although there were slight fluctuations in different weeks, mean scores related to declarative knowledge were observed to decrease dramatically in the second and eighth weeks. Some students explained the reason as follows: S16 "The question seemed challenging", S12: "At first, I had no idea about what to do".

In addition, the development of procedural knowledge was at average level. The explanations of the students who got low scores expressed that the programming problem was challenging and they could not clearly follow the correct solution steps. For instance, S13 stated that; "My Steps: I am planning to reach samples on internet and follow them". This was assigned as "Needs Improvement" category as he mentioned his plan in the process instead of noting each step down in a row. S9 whose explanation was assessed as the "Advanced" category expressed that: "I make some steps in my mind even though they are not very clear: Determining the number of the variables and the processing to be made, investigating whether it could be made without using loop or strings, writing a draft code on a paper, writing on computer." Given that the student explained each step clearly, directly and correctly, the explanation was evaluated under the category of "Advanced".

In the conditional knowledge strategy, students' answers were about getting support. For instance, S12 clearly explained as "The help I will get: I will get help about understanding the structure of Pascal's triangle, and arranging the rows and columns" and it was evaluated as "Advanced". Similarly, S10 clearly as addressed that "The help I will get: I need to learn how to use Timer" and indicated the help s/he would need in another week and clearly talked about it. It was also evaluated as "Advanced".

As the problems became difficult in the last weeks, an acceptable increase was observed in spite of the decrease in planning scores. Planning scores were decreased slightly in week 7. In this line; S1 could not express her purpose clearly by "My purpose: Writing the program as quick as possible. The sources: blogs on internet" and mentioned the sources shallowly. Thus, the explanations of S1 were evaluated as "Acceptable".

Monitoring strategy improved for 10-week of research; however, a decrease was observed between the weeks 5 and 8 (Fig. 4). S8 responded the items 5 and 8 used to assess the decreasing monitoring strategy; "Using the rows and columns in loop. I benefited from the sources when I had difficulty. I checked the program following each item I wrote". This answer was not adequate and assigned as "Needs Improvement".

It was figured out that evaluation strategy scores differed for the whole process. There were drastic decreases in Weeks 2 and 8. Among the students who experienced the decrease, S4 was evaluated under "Needs Improvement" category with the explanation:

> S4: "I am good at writing a program yet I have some problems with the errors. Unfortunately, the problem was not easy. I could not manage to fix the errors of the program. Whenever I met an error, I thought I could not continue anymore. "

S4 explained the reason for this decrease with the fact that he could not fix the errors. S14 who was evaluated as "Advanced" in evaluation strategy explained the background of his success as "My only strength is being a 'researcher' and thanks to it I could assimilate my weaknesses". Similarly, S14 was evaluated in "Advanced" category at the same week due to the fact that she expressed how she could integrate other fields as follows: "I could design similar small-scale games just for fun and play them with my friends".

The Observed Changes in Programming Performances

This study was directed to reveal the programming performances in the given week. The value of the students' answers on weekly worksheets in the ten-week course was evaluated according to the SOLO categories.

The development of the students' programming performances during the 10-week of research is illustrated in Fig. 5.

In the first week of the research, according to the SOLO category, the answers of the students to the problems are in 56% prestructural, 31% unistructural and 13% multistructural categories. In the first week, there were not answers under the categories of Relational and Extended Abstract. In this week, S4 is under prestructural category with his answer; "I need to make description, and to use button, textbox and Label. I remember them in general; however, I think there will be some parts I will need them". Even though this student was aware of the structures he would use, he could not express knowledge of the use of these structures clearly. In the second week, the results showed that 57% of the answers to the problems in the worksheet are in unistructural category. While there were not any answers under the Relational category the previous week, 7% of the answers were under this category in the second week. However, there were not any answers under Extended Abstract category.

In the second week, the answer by S3 was just explaining how he could write the program:

"First of all, it is necessary to know how to explain the variable. First, the variables should be defined and the process should be started with "if else" commands. It is vital to present the cause and effect relationship."

Since she did not mention about her plan for the process or where to integrate what she learned, the answer was evaluated under unistructural category.

In the third week, most of the answers in the worksheet were in unistructural and multistructural categories with the rate of 37%. There was a slight increase in the num-

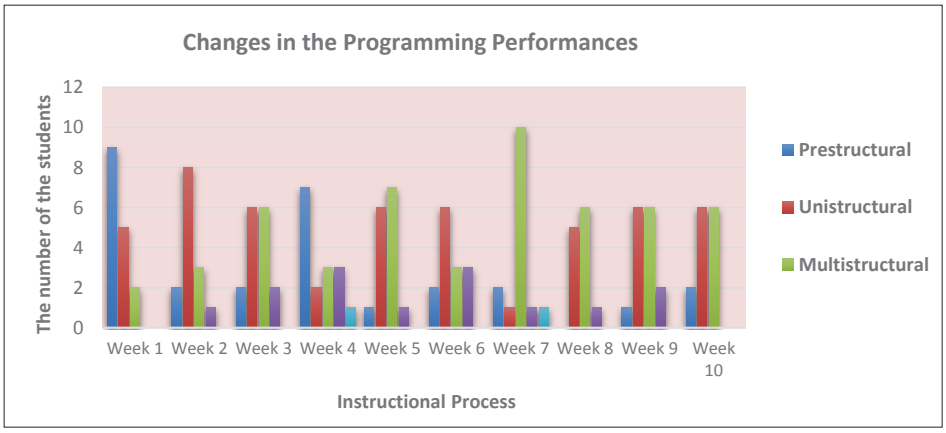


Fig. 5. The development of programming performances during ten weeks.

ber of the students under the relational category for this problem. There was no answer for the category of extended abstract in this week. S1 was evaluated under multistructural category because he properly explained most of the method he would use while writing a program and how to use it. His answer was:

> "I think I should learn how to choose random numbers with Random, match the numbers with the images, present the matching image as the selection of the computer. I think within the context of the logic of Algorithm and put the processing into order in my mind, and finally I apply it on the program. I have gone over my Random knowledge."

In the fourth week, most of the answers to the problems in the worksheet were in prestructural category with the rate of 44%. 12% of the answers were under the unistructural category while 19% of the answers were under multistructural and relational categories. Compared to the previous weeks, the number of the students in the prestructural category decreased and the ones in the relational category increased. In this week, there were answers in each SOLO category and the rate of the students' answers in extended abstract was 6%. In this week, S8 was evaluated under prestructural category with his irrelevant explanation; "I guess I will use 'while loop'".

In the fifth week, most of the answers to the problems in the worksheet were in multistructural category with the rate of 47%. 40% of the answers of that week were under the unistructural category. On the other hand, there was no answer evaluated in the category of extended abstract in this week.

In this week, S12 provided a clear disclosure about the code to use and the steps she would take.

> "Firstly, what is Pascal's triangle and for which rule is formed? The ability to use string, to return string value according to the logic of Pascal's triangle, form the structure of Form. Describe a for loop which controls the values submitted to Textbox. Arrange the columns and rows in the described loop with nested for loop, and operate. This helped me to improve my programming skills. I have no idea where to use."

In the aforesaid rubric, such an explanation is under the category of multistructural.

In the sixth week of the research, most of the answers to the problems in the worksheet were in unistructural category. 14% of the answers were under the prestructural category while 21% of the answers were under multistructural and relational categories.

In this week, S9 was assigned in unistructural category because he could express a part of the codes to use while writing a program though it was not clear.

His explanation was as follows: "I need to study on strings again. Moreover, I do not know how to split the words by using the spaces, I will search for it. I will use for loop and strings. I learned split command and the logic of these strings."

In the seventh week, most of the answers (67%) to the problems in the worksheet were assessed in multistructural category. As an example; S1 answered the questions of the week as follows:

"I should be able to define text strings. Firstly, I made the strings of the numbers in digits. Then, I got the number in digit and got it from the string. Consequently, I learned writing a program like this". As the student expressed the necessary codes needed to write the program, and where and how to use them, his/her explanation was evaluated under multistructural category.

In the eight week, half of the answers (50%) to the problems in the worksheet seemed to be in multistructural category. 42% of the answers were in unistructural while 8% percent of them were in relational category. There were not any answers in prestructural and extended abstract categories of SOLO categories. S4 who was evaluated as multistructural category with his following answer:

"Printing image on the screen, adding text characters into the string and controlling them. I know how to print an image on screen and I could add text characters into the strings yet I have no idea about controlling them. I clipped the images and created the design in the program but I do not know what is next. I learned to print the images and arranging its visibility. I could perform various applications with this." .The student was evaluated under this category given that the student thoroughly wrote the steps to follow program writing process, and the codes.

In the ninth week, most of the answers to the problems in the worksheet seemed to be in multistructural category (40%) and in unistructural (40%) category. There was no answer in the category of extended abstract. S15 who was evaluated as multistructural category answered the question as: "I need to find the squares in chessboard. We add up the exponents of each square. I have the necessary information. I do not think I will need it. I will find the number of the squares with for loop. I will make the exponents of each square add in a different for. It is correct. I may reach it. I contribute nothing for this problem."

The student was evaluated under multistructural category since she could not mention how to use this new knowledge in other fields although she properly explained the codes to use and the steps to take.

In the tenth week, most of the answers to the problems in the worksheet seemed to be in unistructural (43%) category and multistructural category (43%). The 14% of the students' answers were in prestructural category. Due to the fact that S14 seemed to know a part of the code by expressing that she would use conditional structure in C, S14 was evaluated under unistructural category with the following explanation;

> "It could be achieved by creating a good logic with If. I know If structure yet I will need the help of my friend since it is complicated. I am progressing by creating If structure and I suppose it is correct and I will be able to manage it. I may use it for fun in the daily life."

To sum up, it is observed that students' answers were in multistructural category (40%–50%) in the whole process. Because the problems became challenging and complicated progressively, the answers of the students were in multistructural category even though there were some decreases in certain weeks. Moreover, almost 50% of

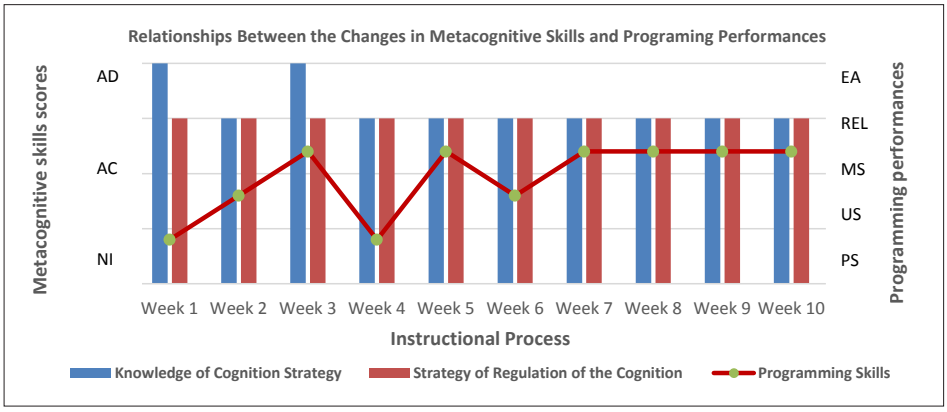
the student answers were in prestructural and unistructural categories. In the relational category, the number of the students ranked between 0 and 9 throughout the process, and in certain weeks no student was in this category. As for the unistructural category, the number of the students ranked between 1 and 8, and there were students in this category each week. In the relational category, there were students only for two weeks, and students could not manage to reach this last step of the SOLO categories in other weeks.

Effect of Metacognitive Strategies on Programming Performances

The average development of students' knowledge of cognition, regulation of cognition and programming performances during the course of ten-week of research is shown in Fig. 6.

Fig. 6 shows the average scores in the dimensions of metacognition and programming performances in the ten-week period. Fig. 6 indicates that the students' performances in multistructural category related to knowledge of cognition and regulation of the knowledge are at reasonable rate. Even though students' metacognitive development in the second and third weeks is at acceptable level, their programming performances are either in prestructural or unistructural category.

Overall, it is seen that the problems are suitable for developing metacognitive strategies in programming. The nature of the related problem has an important role in the development of both cognitive knowledge and cognitive regulation strategies, and those students' cognitive potentials and problem solving experiences have significant effects on their metacognition development.



NI (Needs Improvement), AC (Acceptable), AD (Advanced)
PS (Prestructural), US (Unistructural), MS (Multistructural), Rel (Relational), EA (Extended Approach)

Fig. 6. The development of metacognition strategies and programming performances.

Discussion and Conclusions

In the higher education programming classrooms, many researchers addressed that learning computer programming requires dealing with challenging problems. Since they do not have enough experiences of the programming conceptual and operational knowledge, they generally fail. In this study, meta-cognitive training via worksheets were aimed to guide students how to solve programming problems and how to organize their knowledge for problem solving. In this study, students' use of metacognitive strategies was observed to be at an average level. It is observed that the knowledge of cognition is developed more than the regulation of cognition, which is in accord with other studies (Martinez, 2010). In this sense, the influence of the nature of the problem and the problem solving background for the metacognitive strategy are the reasons for the influence of the knowledge of cognition and regulation of cognition strategies in this study.

The results of the study demonstrated that the metacognitive training provided to the students helped them in improving their learning performance in computer programming. With the contribution of the metacognition strategies in programming, the students' programming performances were developed at multistructural level. In the various studies conducted within this context, a positive correlation was found between using metacognitive strategies and programming performances (Antonietti, Ignazi, and Perego, 2000; Carlson, and Bloom, 2005). In the weeks that knowledge of cognition and regulation of cognition strategies developed more, programming performances also developed as the level of relational and extended approach categories**.**

In the current study, we found that enhancing metacognition contributed to create algorithms more easily allowing various solutions to develop. In this process, it is supposed that the instructions given through the metacognition strategies, thinking about the problem before solving the problem, and making plans made it easier to solve the problems. As Teong (2003) concluded, making decisions based on meta-cognitive strategies are signs of better problem-solving skills. The similar positive effect of the use of metacognitive strategy on problem solving skills has been addressed by various studies (An and Cao, 2014; Goldberg and Bush, 2003; Kapa, 2001; Kock and Harskamp, 2014).

The development in programming performances and metacognition strategies decreased relatively in the last weeks. One reason for this may be that the advantages of metacognitive strategies cannot be enabled when the problems are more complicated. In this research, the fact that students needed to use both logical and mathematical operations as well as their programming knowledge sometimes negatively affected the development of programming performances. Towards the last weeks, the problems become more challenging and sophisticated. One other finding was that students learned to think about the solutions first. The metacognitive support might have supported or sometimes directed them to think about the scope of the problem first rather than the solution.

The SOLO taxonomy was employed to investigate the development of programming performances in this study. The use of SOLO taxonomy allowed the detailed examination of the development of programming performances. Some researchers also assert that SOLO taxonomy corresponds to the nature of programming (Lister *et al.*, 2006; Sheard *et al.*, 2008).

The study also has some certain limitations. First, this study was a small-scale study; a further study is recommended with a larger group to provide different analysis techniques for additional evidence. Second; the study was purposefully limited to 10 problems and focused on the potential of worksheet of metacognitive development related with the problems. However, the problems which were constructed to have a progressive structure of programming knowledge allowed the researchers to analyse the effect of the intervention easily.

To conclude, supporting metacognition via worksheets positively influenced the development of the metacognitive strategies and programming performances. Similar to the other fields, mastering the science of cognition in programming field requires time. Enriching the problem solving sessions via metacognitive interventions provided opportunities for the students to transfer their knowledge in other programming contexts as well as providing the metacognitive knowledge concerning how to enhance their programming skills.

Recommendations

The results of this study indicate that improving the metacognitive knowledge can facilitate programming learning and enhance programming performances in various extents. In line with this result, some recommendations which might be applied in practice are presented below.

- While teaching programming, metacognition strategies might be utilized to help students easily understand the complicated nature of programming.
- It is important that the tools used to develop metacognitive strategies for programming instruction include specific strategies about how to teach within the context of the given content.
- Due to the structure of the programming languages, instead of providing a summative evaluation, SOLO taxonomy can be used to evaluate the programming performances when solving problems.
- In this study, metacognition strategies were provided via the worksheet. Future research might take teaching metacognition strategies via other instruments.
- The nature of the programming problems affected the development and awareness of the metacognitive strategies. Therefore, the relationship between the nature of programming and the development of metacognitive strategy can be investigated in future studies with this point of view.

We hope that the findings of this study would provide implications for programming instructors who desire to provide better programming experiences through metacognitive interventions.

References

An, Y.J., Cao, L. (2014). Examining the effects of metacognitive scaffolding on students' design problem solving and metacognitive skills in an online environment. *Journal of Online Learning and Teaching,* 10(4), 552.

Antonietti, A., Ignazi, S., Perego, P. (2000). Metacognitive knowledge about problem-solving methods. *British Journal of Educational Psychology*, 70, 1–16.

Apiola, M., Tedre, M. (2012). New perspectives on the pedagogy of programming in a developing country context. *Computer Science Education*, 22(3), 285–313.

Carlson, M.P., Bloom, I. (2005). The cycle nature of problem solving: An emergent multidimensional problem-solving framework. *Educational Studies in Mathematics*, 58, 4575.

Chalmers, Christina , Nason, Rodney A. (2005). Group metacognition in a computer-supported collaborative learning environment. In: Looi, Chee-kit,Jonassen, David H., & Ikeda, M. (Eds.), *Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences: Sharing Good Practices of Research, Experimentation and Innovation.* IOS Press, Amsterdam, 35–41.

Chen, G.D., Li, L.Y., Wang, C.Y. (2012). A community of practice approach to learning programming. *Turkish Online Journal of Educational Technology*, 11(2), 15–26.

De Kock, W.D., Harskamp, E.G. (2014). Can teachers in primary education implement a metacognitive computer programme for word problem solving in their mathematics classes? *Educational Research and Evaluation*, 20(3), 231–250.

Derry, S.J. (1992). Adaptative learning environments − foundations and frontiers. In: M. Jones and P.H. Winne (Eds.), *Metacognitive Models of Learning and Instructional Systems Design*. Springer−Verlag Berlin Heidelberg: Nato ASI Series Books, pp. 257−286.

Flavell, J.H. (1976). Metacognitive aspects of problem solving. In: L.B. Resnick (Ed.), *The Nature of Intelligence.* Hillsdale, NJ: Lawrence Erlbaum, 231–235.

Ford, J.L. (2007). *Programming for the Absolute Beginner.* Boston, MA, USA: Course Technology.

Goldberg, P.D., Bush, W.S. (2003). Using metacognitive skills to improve 3rd graders' math problem solving. *Focus on Learning Problems in Mathematics*, 5(10), 29–48.

Gundurao, H.K., Manjunath, N.S., Nachappa, M.N. (2010). *Computer Technology and Computer Programming*. Mumbai, IND: Global Media.

Hartley, K. (2001). Learning strategies and hypermedia ınstruction. *Journal of Educational Multimedia and Hypermedia*, 10(3), 285–305.

Hartman, H.J. (2001). Teaching metacognitively. In: H.J. Hartman (Ed.), M*etacognition in Learning and Instruction: Theory, Research and Prac*tice. Dordrecht: Kluwer Academic Publishers, Boston, 149–172.

Helminen, J., Malmi, L. (2010, October). *Jype-a Program Visualization and Programming Exercise Tool for Python.* Paper presented at the 5th international symposium on Software visualization. New York, USA, 153–162.

Kafai, Y., Resnick, M., MaLoney, J. (2009). Scratch: Programming for all. *Communications of the Acm*, 11(52), 60–67.

Kapa, E. (2001). A metacognitive support during the process of problem solving in a computerized environment. *Educational Studies in Mathematics,* 47(3), 317–336.

Kayashima, M., Inaba, A., Mizoguchi, R. (2004). What is metacognitive skill? Collaborative learning strategy to facilitate development of metacognitive skill. In: L. Cantoni., C. McLoughlin (Eds.), *Procedia of World Conference on Educational Multimedia, Hypermedia and Telecommunications*. Lugano, Switzerland, 2660–2665.

Lee, C.B., Teo, T., Bergin, D. (2009). Children's use of metacognition in solving everyday problems: An initial study from an Asian context. *The Australian Educational Researcher*, 36(3), 89–102.

Lin, X., Schwartz, D.L., Hatano, G. (2005). Toward teachers adaptive metacognition. *Educational Psychologist*, 40(4), 245–255.

Linn, M., Dalbey, J. (2013). Cognitive consequence of programming instruction. In Soloway, E., and Spohrer, J.C. (Eds.), *Studying the novice programmer*. Psychology Press, Hillsdale, 57–81.

Linn, M.C., Clancy, M.J. (1992). The case for case studies of pro-gramming problems, *Communications of the ACM*, 35, 121–132.

Lister, R., Simon, B., Thompson, E., Whalley, J.L., Prasad, C. (2006). *Not seeing the forest for the trees: novice programmers and the SOLO taxonomy.* Paper presented at the 11th Annual Innovation and Technology in Computer Science Education, Italy, 118–122.

Mann, M., Treagust, D.F. (1998). A pencil and paper ınstrument to diagnose students conception of breathing, gas exchange and respiration. *Australian Science Teachers Journal*, 44(2), 55–59.

Martinez, R.E. (2010). *The Use of Metacognitive Tool in an Online Social Supportive Learning Enviroment: An Activity Theory Analysis* (Unpublished doctoral thesis). University of Missouri, St. Louis, USA.

McCormick, C.B. (2003). Metacognition and learning. In: W.M. Reynolds, and G.E. Miller (Eds.), *Handbook of Psychology: Educational Psychology*. Hoboken: Wiley, 79–102.

Park, O. (1992). Instructional applications of hypermedia: Functional features, limitations, and research ıssues. *Computers in Human Behavior,* 8, 259–272.

Patton, M.Q. (2014*). Nitel Araştırma ve Değerlendirme Yöntemleri*. (M. Bütün & S.B. Demir, Çev). Ankara: Pegem Akademi.

Robins, A., Rountree, J., Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education,* 13(2), 137–172.

Rum, S.N.M., Ismail, M.A. (2014). Usability evaluation of metacognitive support system for novice programmers. *Asian Journal of Education and e-Learning*, 2(5). Retrieved November 19, 2018 from http://ajouronline.com/index.php/ AJEEL/article/view/1819.

Salomon, G., Perkins, D.N. (1989). Rocky roads to transfer: Rethinking mechanisms of a neglected phenomenon. *Educational Psychologist*, 24, 113–142.

Schraw, G., Moshman, D. (1995). Metacognitive theories. *Educational Psychology Review,* 7(4), 351–371.

Sheard J., Carbone A., Lister R., Simon B., Thompson E., Whalley J.L. (2008). Going SOLO to assess novice programmers. *ACM SIGCSE Bulletin.* 40(3), 209–213.

Storey, S. O. (2004). *Teacher Questioning to Improve Early Childhood Reasoning* (Unpublished doctoral thesis). University of Arizona, Tucson, Arizona.

Tan, K.C.D., Goh, K.N., Chia, S.L., Treagust, D.F. (2002). Development and application of a two-tier multiple choice diagnostic instrument to assess high school students' understanding of inorganic chemistry qualitative analysis. *Journal of Research in Science Teaching*, 39(4), 283–301.

Treagust, D.F. (1988). Development and use of diagnostic tests to evaluate students' misconception in science. *International Journal of Science Education*, 10(2), 159–169.

Whalley, J.L., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P., Prasad, C. (2006). An Australasian study of reading and comprehension skills in novice programmers, using the bloom and SOLO taxonomies. In: *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*. Australian Computer Society, Inc., pp. 243–52.

Veenman, M.V.J. (2005). The assessment of metacognitive sıklls: What can be learned from multi-method designs?. In: B. Moschner, C. Artelt (Eds.) *Lernstrategien und Metakognition: Implikationen für Forschung und Praxis*. Berlin: Waxmann, 75–97.

Yin, R.K. (2003). *Case study research: Design and methods.* (3rd ed.). Thousand Oaks, CA: Sage.

Zapu ek, M., Rugelj, J. (2013). *Applying ideas from intelligent tutoring systems for teaching programming in game based learning.* Paper presented at the 7th European Conference on Games Based Learning. Porto, Portugal, 11–20.

Ü. Çakiroğlu PhD is a full professor of Computer and Instructional Technologies at Trabzon University, Turkey. His research interests include instructional technologies and computer science education. His academic specialty is instructional technologies, learning analytics, artificial intelligence in education and methods for teaching programming and robotics.

B. Er is a PhD student at Computer and Instructional Technologies at Trabzon University, Turkey. Her research interests include programming instruction and instructional technologies, and her academic speciality include instructional design, and technology integration.