Reflective Thinking Skills on Computational Thinking And Problem Solving As A Predictor of Self-Efficacy of Informatics Teacher Candidates on Programming

Article · April 2018

DOI: 10.29299/kefad.2018.19.009

CITATIONS
0

READS
235

3 authors:

Serdar Ciftci
Adnan Menderes University
11 PUBLICATIONS **20** CITATIONS
SEE PROFILE

Meltem Cengel
Adnan Menderes University
15 PUBLICATIONS **24** CITATIONS
SEE PROFILE

Muhammed Paf
Education
1 PUBLICATION **0** CITATIONS
SEE PROFILE

http://kefad.ahievran.edu.tr

Ahi Evran University
Magazine of the Faculty of Education
Kırşehir

ISSN: 2147 - 1037

Reflective Thinking Skills on Computational Thinking and Problem Solving as a Predictor of Self-Efficacy of Informatics Teacher Candidates on Programming

Serdar Çiftci
Meltem Çengel
Muhammed Paf

DOI:..... **Article Information**

Uploading:25/08/2017 Correction:01/12/2017 Approval:14/03/2018

Summary

Self-efficacy is the belief in an individual's own abilities and capacity. If the person believes in their capacity to achieve in a given area, higher levels of motivation and attention to detail will be positively reflected in their performance. Self-efficacy related to programming can be understood as the belief in one's own capacity to attain a higher level of motivation and an increase in self-worth while writing a program. The purpose of the research is to examine perceptions of self-efficacy for prospective computer programming and instructional technology teachers, and how different perceptions of self-efficacy can be predicted through an analysis of demographic variables, reflective thinking skills on computational thinking and problem solving. Data for the survey were gathered from teacher candidates using a questionnaire containing personal information and demographic information and further consisted of four categories: 1) the Self-efficacy Scale for Programming developed by Ramalingam and Wiedenbeck (1998); 2) a customized version of this scale for Turkish sites by Altun and Mazman (2012); 3) the Reflective Thinking Scale for Problem Solving developed by Kızılkaya and Aşkar (2009); 4) Computational Thinking Scale developed by Korkmaz, Çakır and Özden (2017). Data were obtained from teacher candidates studying Computer Education and Instructional Technology Education Department at four different universities in Turkey. Multiple regression analysis was used in the analysis of the data. According to the results of the analysis, the independent variables were significantly predicted by the self-efficacy value of the program ($F(3, 99) = 24.880$, $p < .001$). The most important contribution to the formation of the model is computational thinking, followed by reflective thinking about problem solving and follow-up computer-related developments. The adapted R^2 value is at .413, indicating that it discloses 41% of the self-sufficiency relating programming with the model.

Key Words: informatics teacher candidates, programming self-efficacy, computational thinking, reflective

Sorumlu Yazar : Serdar Çiftci, Assist. Prof. Dr., Adnan Menderes University, Turkey, sciftci@gmail.com, https://orcid.org/0000-0001-5282-1861
Meltem Çengel, Assist. Prof. Dr.Adnan Menderes University, Turkey, meltemcengel@gmail.com, https://orcid.org/0000-0002-0255-4600
Muhammed Paf, Curriculum and Instruction Dept. Grad. Stu., muhammedpaf@gmail.com
This study has been presented at XV. European Conference on Social and Behavioral Sciences.

thinking about problem solving

Introduction

Current research indicates that individual and internal resources such as the concept of self-efficacy have gained importance among factors affecting the participation and performance of individuals (Chen, 2017). Self-efficacy is generally the belief of an individual in how they will perform his or her task (Huffman, Whetten, & Huffman, 2013). According to Bandura (1994), self-efficacy is not a concept that only makes "prophecies" about performance level. Self-efficacy stimulates the components of motivation and triggers the cognitive skills and effort required to perform the task (Kher, Downey, & Monk, 2013). Davidsson, Lazron and Ljunggren (2013) further indicate that an individual's belief in self-efficacy internally supports the ability to specify activities needed to accomplish a task and in providing mechanisms that assist in managing difficulties and performance anxieties.

Present research examining computer and programming skills (Bargury et al., 2012; Bers, Flannery, Kazakoff, & Sullivan, 2014; Grout & Houlden, 2014; Jones, Mitchell, & Humphreys, 2013), offer clear examples of the need to explore individual and internal processes underlying both learning and performance outcomes. This research supports the concept of self-efficacy as being extremely important in learning computer skills (Karsten, Mitra, & Schmidt, 2012). Likewise, during the learning of these skills, a cause of failure in the programming lessons can be linked to student perceptions of low self-efficacy. Specifically, students who struggle with learning essential programming skills (Askar & Davenport, 2009) has led Igbaria and Iivari (1995) to suggest that low self-efficacy may be a contributing factor of underperformance in the technology classroom.

When studies on self-efficacy related to learning computer programming skills are examined in the context of Turkish universities, clear differences can be discerned between the self-regulatory strategies and overall achievements of students who are pursuing different curricular paths (Haşlaman & Aşkar, 2007). Differences between self-efficacy and attitudes toward programming (Özyurt & Özyurt, 2015; Yağcı, 2016) include, students who are specifically studying computer programming in the Computer and Instructional Technology Education program (BÖTE), as compared to students who are attending different departments while taking computer programming courses (Gezgin & Adnan, 2016; Mazman & Altun, 2013). Similar research by Psycharis and Kallia (2017) examined the effects of computer programming skills on the role self-efficacy may have for high school students, particularly in the asking new questions, solving mathematical problems. Additional research by Maddrey (2011) examined the relationship between positive effects of self-efficacy and problem-solving curriculum delivered to computer science students.

In addition to supporting existing research, the main purpose of this study is to reveal and examine potential predictors of self-sufficiency in the curricular design for computer science courses. Here we examine how reflective thinking, computational thinking and certain demographic variables related to problem solving can provide a more nuanced understanding of the relationship between student perceptions of self-efficacy and positive learning and performance outcomes for computer programming students.

Theoretical Framework

Reflective Thinking on Problem Solving

Writing a computer program includes complex and challenging tasks such as abstraction, mathematical logic, testing, debugging and other forms of solving problems (Saeli et al., 2011). This assertion is supported by Soloway (1993), who indicates that an effective process for writing a computer program requires individuals to use problem solving / designing / thinking strategies. To effectively program their assignments, students must first mathematically find out how the problem will be solved, then how to transfer it to the computer, and finally how to correctly translate it to the software language (Psycharis & Kallia, 2017). The program writing process also requires in-depth reading and cognitive awareness skills about how the problem can be effectively solved. For this reason, program writing involves reviewing decisions and actions continuously to improve the quality of the program and producing alternative methods that may lead to more eloquent result(s) (Kalelioğlu, 2015). Govender et al. (2014), suggests that some skills required in the program writing process are not taught explicitly, therefore teaching problem solving, one of the skills required, should not only include the presence and practice on problems, but should also include teaching the problem-solving process.

The problem-solving process can easily become unclear, particularly in the case where there are many possible solutions and evaluations for a given example and/or the problem is not very well structured (ill-structured problems). In these instances, reflective thinking is required because reflection includes the stages of process control, monitoring, and gaining experience (Lee, Teo, & Zealand, 2011). The concept of reflection contains different connotations in different contexts, but in this study, reflective thinking is limited to the problem-solving context (Kizilkaya & Ashkar, 2009). Here, reflection includes an analysis of the decision-making process about what is happening, that is, what is the problem-solving process required to ensure the intended end-result of the computer programming assignment. Reflective practices are also important because it allows the student to take a step back, giving them new or additional ideas of how they can solve a problem and helps them to understand and appreciate how useful reflective strategies are for consistently finding an optimal

solution to any given problem (Kalelioğlu, 2015). The practice of reflective thinking is significantly different from the processes applied to other forms of problem-solving contexts, in terms of the scope of suspicion of the situation, hesitation, confusion, mental difficulty, inquiry, finding material to resolve doubts, et cetera.

There are many studies focusing on the relationship between programming processes, problem solving and reflective thinking (Fessakis, Gouli, & Mavroudi, 2013; Liao & Bright, 1991). For example; Bergin, Reilly and Traynor (2005) indicate that students who employ strategies involving metacognitive and administrative processes are more successful in the programming process. While Akcaoglu and Koehler (2014) indicate that problem-solving skills of students participating in game-design learning groups show a significant level of problem-solving success compared to those who do not participate. Additionally, Ke (2014) remarks that involvement in game programming contributes to student success in mathematics-related concepts and problem-solving skills. The study of Psycharis and Kallia (2017) is related to whether the program has a significant influence on student self-efficacy in asking critical questions, problem solving, and mathematics. According to this study, programming on a computer has a significant effect on self-efficacy, but does not have a significant effect on problem solving.

Computational Thinking

Currently there are competing definitions within Turkish literature surrounding the notion of computational thinking. The following demonstrates various attempts to employ computational thinking as a generative framework for analysis: Computational thinking (Şahiner & Kert, 2016), Data-Operational thinking (Demir & Seferoglu, 2017), Computational thinking (Çatlak, Tekdal & Baz, 2015), Computational thinking (Sayın & Seferoğlu, 2016), Calculative thinking (MEB, 2017), Computational thinking (Aldağ & Tekdal, 2015; Şahiner & Kert, 2016) and Data-Operational thinking (Barut, Tuğtekin & Kuzu, 2016; Kalelioğlu, Gülbahar, Akçay and Dogan, 2015). Clearly, competition to define, and thereby control, the nomenclature of this typology is in and of itself an adequate description of the importance that computational thinking has in current educational thought and practice. In addition to this, expressions like software development, coding, program writing, and writing computer programs are interchangeable with the concept of "programming". Understanding the shifting meanings of computational thinking currently competing for stability and clarity in the field, our use of computational thinking is a multidimensional skills problem solving and adapting this skill to the computer science.

As indicated above, one of the most important variables associated with the ability to write computer programs is the concept of computational thinking (Román-González, 2015). Introduced by

Wing (2006), computational thinking includes ways of solving problems, designing systems and understanding human behaviors. There are a variety skills that comprise or involve the conceptualization of, "knowing about computers," or the various practices implied in the notion of "programming." This complex of skills involves knowledge of applications and aspects of problem-solving dealing with loops, repetitions, and conditions that occur in the programming process, as well as an understanding that the point of view of the programmer may sit in consonance or dissonance to the point of view of the user in the technological world (Lye & Koh, 2014).

Computational thinking, then, is not simply a concept that only computer scientists use (Kim, Kim, & Kim, 2013)—Kafai and Burke (2013) mention that technological tools are not needed to talk about the concept of computational thinking—as elements can also be found within other knowledge domains. For example, Wing (2006) describes computational thinking as form of critical thinking and as the application of existing knowledge to the solution of complex problems in mathematics, science and generally FETEMM (Science, technology, engineering and mathematics-STEM). According to Aho (2012), computational thinking can be formulated as partitions of problems, abstraction, logical thinking, algorithm creation and debugging. In sum, Computational thinking can be defined as a concept that includes problem solving strategies and computer science structures such as abstraction at different hierarchical levels, algorithmic thinking, automatization, segmentation, modeling, pattern subtraction, repetition, scaling and symbolic representation, (Grover & Pea, 2013; Korkmaz, Çakir, & Özden, 2017; Ni & Guzdial, 2012; Psycharis & Kallia, 2017; Werner & Denning, 2009).

However, there are also studies that relate this concept to analytical and problem-solving abilities, as well as habits, attitudes that arise when errors and problems are encountered specific to the adoption of approaches used in computer science (Bers et al., 2014).

While not focusing on the relationship between self-efficacy and cognitive thinking about directly programming on the computer, Pellas (2014) found in his study that the students' participation in the course was significantly predicted by computer self-efficacy, metacognitive self-aligning strategies and self-esteem and that there is a significant relationship between these independent variables. In an experimental study on programming with robotics, Jaipal-Jamani and Angeli (2017) found that self-efficacy beliefs in robotics is associated with increased field knowledge levels and levels of computational thinking of teacher candidates. In another study (Psycharis & Kallia, 2017), it is mentioned that the teaching of programming, when enfolded into the mathematics curriculum, has an effect on student math self-efficacy. Lastly, in their qualitative analysis of curricular and program design that link programming with computational thinking, Pellas and Peroutseas (2016) found increased proficiencies in both programming skills and how students learn to think about computer based problem sets.

Method

Participants

The survey has 166 participants. Ninety-three of them are male (56%), seventy-three are female (44%). The participants attended four different universities, namely the Adnan Menderes University (n=48, 28.9%), the Muğla Sıtkı Koçman University (n=14, 8,4%), the Süleyman Demirel University (n=53, 31,9%) and the Ereğli Faculty of Education of the Konya Selçuk University (n=50, 30,1%). 48,2% (n=80) of the participants are in the second year, while 22,9% (n=38) are in the third year and 28,9% (n=48) are in the fourth. 24,1% (n=40) has graduated from the Anatolian High School, while 65,7% (n=109) has graduated from the Vocational High School or the Technical High School, 10,2% (n=17) has graduated from other high schools. Sixty participants (96.4%) have a computer, while six participants (3.6%) do not have a computer. Seventeen students (14.8%) are following computer science related developments, while 98 students (n=59.0) are not following these developments.

Tools

A 10 point personal information form for students to determine some demographic and computer usage habits, a self-efficacy scale for programming, a reflective thinking scale for problem solving and a computational thinking scale was applied in the analysis tool set. The adaptation of the Self-Efficacy Scale for Programming developed by Ramalingam and Wiedenbeck (1998) to Turkish was done by Altun and Mazman (2012) in order to demonstrate the self-efficacy of using C ++ program upon university students. A two-factor structure consisting of nine items has been achieved as a result of the adaptation study. The internal consistency coefficient of the scale consisting of two dimensions, namely "complex programming tasks" and "simple programming tasks", was found, which explains 80,814% of the total variance, and a model confirmatory factor analysis was also conducted for this structure.

The Reflective Thinking Scale on Problem Solving developed by Kızılkaya and Aşkar (2009) is composed of 14 items and 3 sub-dimensions. The three basic dimensions are the questioning, evaluation and reasoning dimensions. An example for the size of an inquiry is the item "I ask myself questions to determine what is given and desired when I read the problem". An example of the evaluation dimension is the item "I'm trying to solve the next problem better through evaluating my solutions over and over again". As an example of the cause dimension, there is the item "When solving a problem, I try to establish the relationship with the result I have found, considering the reasons for the actions I have made". Scale items were rated according to the 5-point Likert scale. The frequency of actions contained in the items are arranged at "Always", "Mostly", "Sometimes", "Rarely", "Never". These levels are graded as Always = 5, Most times = 4, Sometimes = 3, Rarely = 2, Never = 1.

Confirmatory factor analysis results of the scale were calculated as GFI = 0,92, AGFI = 0,89, NNFI = 0,93, CFI = 0,95, RMSR = 0,08, RMSEA = 0,071.

The scale developed by Korkmaz, Çakır and Özden (2017) and applied on university students consists of twenty-two items and five factors. The scale is a 5-point Likert scale consisting of 22 items that can be grouped under five factors. Confirmatory factor analysis was performed to confirm the factor structures of the scale. The scale consists of creativity (4 items), algorithmic thinking (4 items), cooperativeness (4 items), critical thinking (4 items) and problem solving (6 items). The total Cronbach alpha reliability score of the scale is .81. The scale explains 56.1% of the total variance.

Analysis

Approximately 20 minutes after the application, the data were transferred to SPSS and multiple linear regression analysis was performed to analyze the data. In order to perform multiple regression analysis, the assumptions to be met, the predicted variables and the dependent variable have to be linear, the faults and debris have to be normally distributed and unrelated to the predictor. After all, one of the most problematic situations is the problem of multicollinearity. Multicollinearity is observed when there is a high correlation between variables, therefore it was decided for the absence of multicollinearity after controlling the tolerance and VIF values (Leech, Barrett, & Morgan, 2005).

Findings and Comments

Multiple linear regression analysis was performed to determine whether it is predictable to follow self-sufficiency on programming, reflective thinking on problem-solving, computational thinking and computer-related publications.

First, descriptive and relational statistics of related variables are presented in Table 1.

Table 1. Mean, Standard Deviation and Relationship Levels of Self-efficacy and Independent Variables related to Programming

Variables	X	SD	1	2	3
Self-efficacy related to programming	41.796	12.371	.563*	.554*	-.365*
1. Computational Thinking	84.796	11.310	-	.585*	-.233**
2. Reflective thinking related to problem solving	54.815	9.295		-	-.293*
3. Following computer related developments	.845	.364			-

* p< .001

** p< .009

According to Table 1, there is significant relationship between programming self-efficacy and computational thinking at a level of .563, between self-efficacy and reflective thinking about problem solving at a level of .554 and between self-efficacy and computer-related developments at a level of -.365 (p <.001). There is significant relationship between computational thinking and reflective thinking about problem solving at a level of .585, between computational thinking and computer-related developments at a level of -.233 and between computational thinking and reflective thinking about problem solving and computer-related developments at a level of -.293.

The regression model containing the relevant variables is summarized in Table 2.

Table 2. Summary of the Multiple Regression Model relating the prediction of Self-efficacy on Programming by computational thinking, reflective thinking about problem solving and follow-up of computer developments

Variable	B	SD	β
Computational Thinking	.377	8.974	.345
Reflective Thinking about Problem Solving	.391	.103	.294
Follow-up of Computer Developments	-6.756	.127	.199
Fixed	-3.856	2.706	

According to the results of the analysis, independent variables are significantly predicting (F (3, 99) = 24.880, p <.001) the self-efficacy value of the program. The three independent variables contribute significantly to the formation of the model. According to the beta values presented in Table 2, the most important contribution to the formation of the model is computational thinking, followed by reflective thinking about problem solving and follow-up of computer-related developments. The value of R2, adapted to the analysis results, is .413, and that is explaining 41% of the self-sufficiency relating programming models.

Discussion and Recommendations

In general, self-efficacy is the belief in the extent to which an individual will perform a skill (Huffman, Whetten, & Huffman, 2013), while self-efficacy in programming is the belief that the individual will be successful while writing the program. Askar and Davenport (2009) explain that low self-efficacy perceptions during the learning process of programming skills, can trigger failures in programming lessons. According to the analysis results made within the research, self-sufficiency in programming is predicted significantly by computational thinking, reflective thinking about problem-solving, and follow-up of computer-related developments. The students' self-efficacy on robotics and computational thinking skills were significantly influenced by the experimental process in the experimental study conducted by Jaipal-Jamani and Angeli (2017) on robotics self-efficacy and computational thinking of elementary school teacher candidates.

As described above, problem-solving skills developed with computational thinking can be transferred to other fields such as mathematics. In his definition, Wing (2006) expresses computational thinking as a way of problem solving, while Maddrey's (2011) study demonstrates there is a close relationship between problem-solving skills and self-efficacy in programming. In this study, it is observed that problem-solving teaching, as a component of a mathematics curriculum, increases students' self-sufficiency in programming. Research investigating the relationship between inquiry skills, problem solving, and mathematical self-efficacy within programming education, indicates that programming education has a significant effect on questioning skills and mathematical self-efficacy and also contributes significantly to problem-solving skills (Psycharis & Kallia, 2017).

However, Kalelioğlu (2015) explained in his experimental study of programming education for elementary school students, that programming skills education has no significant influence on reflective thinking about problem solving. Another conspicuous finding in the research results is that there can also be a negative relationship between self-efficacy in programming and concurrent follow-up of computer science related developments. According to this result, it can be concluded that individuals with high self-efficacy skills in programming are less likely to follow developments in the area. This may be related to the concept of biasedness that some studies relating the concept of computer self-efficacy state that self-efficacy is important as well as this self-evaluation based concept is far from neutrality. Neutrality is associated with more or less perception of its performance than it is (Pajares & Graham, 1999). In general, perceiving your performance and self-efficacy a little better may increase the effort and the continuity of this effort (Bandura, 1986; Pajares, 1997). However, individuals who perceive too much on self-performance may not be able to change the way they work or develop themselves more (Zimmerman, Bonner, and Kovach, 1996). Students who engage in more reflexive practices are able to evaluate their performances more accurately (Zell and Krizan, 2014). Likewise, Ehrlinger et al. (2008) relate that the more experience people receive as part of a reflective

assessment strategy on any subject also receive more feedback on their performance and cognitive capacities.

For example, observations drawn from this study confirm that accuracy and bias in computer self-efficacy can be linked to the level of prior experience in programming that students bring to the learning task. The more experience students underestimated their self-efficacy as reported in students' accuracy and bias towards computerized self-efficacy reports. Nevertheless, students with higher cognitive abilities seem to perform more consistently and accurately with their performance in self-assessment scales (Aesaert, Voogt, Kuiper, van Braak, 2017). From here, it can be considered that students who follow the developments more often find their efficacy lower.

Based on the research findings, it may be suggested that reflective assessment and computational thinking skills related to problem solving should be included in the training programs as part of the programming courses applied at different levels. Furthermore, studies should be conducted to determine other variables that predict self-efficacy for programming, and, as another research issue, there is a need for studies that examine actual performance on programming, self-efficacy in programming, and other variables that affect the relationship between these two concepts.

References

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, *55*(7), 833–835. http://doi.org/10.1093/comjnl/bxs074

Akcaoglu, M., & Koehler, M. J. (2014). Cognitive outcomes from the Game-Design and Learning (GDL) after-school program. *Computers and Education*, *75*, 72–81. http://doi.org/10.1016/j.compedu.2014.02.003

Aldağ, H., & Tekdal, M. (2015). Gender differences in computer use and programming teaching. Proceeding of the 1st International Cukurova Women's Studies Congress, 236-243.

Altun, A., & Mazman, S. G. (2012). The validity and reliability study of the Turkish form of the self-efficacy perception scale for programming. *Journal of Measurement and Evaluation in Education and Psychology*, 3 (2), 297-308.

Askar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java programming among engineering students. *The Turkish Online Journal of Educational Technology*, *8*(1), 1303–6521. Retrieved from http://files.eric.ed.gov/fulltext/ED503900.pdf

Bandura, A. (1986). *Social foundations of thought and action: A social cognitive theory.* Englewood Cliffs, NJ: Prentice Hall.

Bandura, A. (1994). Self-efficacy. In V. S. Ramachaudran (Ed.), Encyclopedia of human behavior (pp. 71–81). New York: Academic.

Bargury, I. Zur, Haberman, B., Cohen, A., Muller, O., Zohar, D., Levy, D., & Hotoveli, R. (2012). Implementing a new Computer Science Curriculum for middle school in Israel. *2012 Frontiers in Education Conference Proceedings*, 1–6. http://doi.org/10.1109/FIE.2012.6462365

Barut, E., Tuğtekin, U., & Kuzu, A. (2016). Outlook on Robot Applications and Computational Thinking Skills. *3rd International Conference on New Trend in Education (ICNTE 2016)*.

Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. *First International Workshop on Computing Education Research*, 81–86. http://doi.org/10.1145/1089786.1089794

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, *72*, 145–157. http://doi.org/10.1016/j.compedu.2013.10.020

Chen, I. S. (2017). Computer self-efficacy, learning performance, and the mediating role of learning engagement. *Computers in Human Behavior*, *72*, 362–370. http://doi.org/10.1016/j.chb.2017.02.059

Çatlak, Ş., Tekdal, M., & Baz, F. Ç. (2015). The Situation of Programming with Scratch Software: A Document Review Study. *Journal of Instructional Technologies & Teacher Education*, *4*(3). Accessed from http://www.jitte.org/article/view/5000163313 on

Davidsson, K., Larzon, L., & Ljunggren, K. (2013). Self-Efficacy in Programming among STS Students. Retrieved from http://www.it.uu.se/edu/course/homepage/datadidaktik/ht10/reports/Self-Efficacy.pdf

Demir, Ö., & Seferoglu, S. S. (2017). New Concepts, Different Uses: An Evaluation of Information-Operational Thinking (pp. 801-830).

Ehrlinger, J., Johnson, K., Banner, M., Dunning, D., & Kruger, J. (2008). Why the unskilled are unaware: Further explorations of (absent) self-insight among the incompetent. Organizational Behavior and Human Decision Processes, 105,98-121.

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

Gezgin, D. M., & Adnan, M. (2016). Investigation of Self-Efficacy Perceptions of the Students of Mechanical Engineering and Econometrics Related to Programming. *Journal of the Kırşehir Faculty of Education of the Ahi Evran University*, 17 (2), 509-525.

Govender, I., Govender, D., Havenga, M., Mentz, E., Breed, B., Dignum, F., & Dignum, V. (2014). Increasing self-efficacy in learning to program: exploring the benefits of explicit instruction for problem solving. *TD The Journal for Transdisciplinary Research in Southern Africa*, *10*(1), 187–200.

Grout, V., & Houlden, N. (2014). Taking Computer Science and Programming into Schools: The

Glyndŵr/BCS Turing Project. *Procedia - Social and Behavioral Sciences*, *141*, 680–685. http://doi.org/10.1016/j.sbspro.2014.05.119

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

Haşlaman, T., & Aşkar, P. (2007). Examination of the Relationship Between Programming Lesson and Related Self-Regulation Learning Strategies and Success. *Hacettepe University Journal of the Faculty of Education*, 32, 110-122.

Huffman, A. H., Whetten, J., & Huffman, W. H. (2013). Using technology in higher education: The influence of gender roles on technology self-efficacy. *Computers in Human Behavior*, *29*(4), 1779–1786. http://doi.org/10.1016/j.chb.2013.02.012

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

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of Robotics on Elementary Preservice Teachers' Self-Efficacy, Science Learning, and Computational Thinking. *Journal of Science Education and Technology*, *26*(2), 175–192. http://doi.org/10.1007/s10956-016-9663-z

Jones, S. P., Mitchell, B., & Humphreys, S. (2013). Computing at school in the UK : from guerrilla to gorilla. *Communications of the ACM*, (April), 1–13. http://doi.org/10.1016/j.compedu.2013.10.020

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

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, *52*, 200–210. http://doi.org/10.1016/j.chb.2015.05.047

Kalelioğlu F, Gülbahar Y, Akçay S, Dogan D. (2014). Curriculum Integration Ideas for Improving the Computational Thinking Skills of Learners through Programming via Scratch.. *7 th International Conference on Informatics in Schools: Situation, Evolution and Perspectives*: İstanbul; 22/09/2014 - 25/09/2014

Karsten, R., Mitra, A., & Schmidt, D. (2012). Computer self-efficacy : A meta-analysis. *Journal of Organizational Ans End User Computing*, *24*(4), 54–80.

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers and Education*, *73*, 26–39. http://doi.org/10.1016/j.compedu.2013.12.010

Kher, H. V., Downey, J. P., & Monk, E. (2013). A longitudinal examination of computer self-efficacy change trajectories during training. *Computers in Human Behavior*, *29*(4), 1816–1824. http://doi.org/10.1016/j.chb.2013.02.022

Kim, B., Kim, T., & Kim, J. (2013). Paper-and-Pencil Programming Strategy toward Computational Thinking for Non-Majors: Design Your Solution. *Journal of Educational Computing Research*, *49*(4),

437–459. http://doi.org/10.2190/EC.49.4.b

Kizilkaya, G., & Aşkar, P. (2009). Improvement of the reflective thinking ability scale for problem solving. *Education and Science,* 34 (154), 82-92.

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, *72*, 558–569. http://doi.org/10.1016/j.chb.2017.01.005

Lee, C., Teo, T., & Zealand, N. (2011). Shifting Pre-service Teachers â€™ Metaeognition Through Problem Solving, *3*, 570–578.

Leech, N. L., Barrett, K. C., & Morgan, G. A. (2005). *SPSS for Intermediate Statistics: Use and Interpretation* (2nd ed.). New Jersey: Lawrence Erlbaum Associates.

Liao, Y.-K. C., & Bright, G. W. (1991). Effects of Computer Programming on Cognitive Outcomes: A Meta-Analysis. *Journal of Educational Computing Research*, *7*(3), 251–268. http://doi.org/10.2190/E53G-HH8K-AJRR-K69M

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

Maddrey, E. (2011). *The Effect of Problem-Solving Instruction on the Programming Self- efficacy and Achievement of Introductory Computer Science Students*. *Information Sciences*. Nova Southeastern University.

Mazman, S. G., & Altun, A. (2013). The effect of introductory to programming course on programming self efficacy of CEIT students. *Journal of Instructional Technologies & Teacher Education*, *2*(3), 24–29.

Ni, L., & Guzdial, M. (2012). Who AM I?: Understanding high school computer science teachers' professional identity. *SIGCSE '12: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 499–504. http://doi.org/10.1145/2157136.2157283

Özyurt, Ö., & Özyurt, H. (2015). A study for determinin computer programming students' attitudes towards progmming and their programming self-efficacy. *Theory and Practice in Education /Articles Journal of Theory and Practice in Education*, *11*(1), 51–67.

Pajares, F., & Graham, L. (1999). Self-efficacy, motivation constructs, and mathematics performance of entering middle school students. *Contemporary Educational Psychology*, 24, 124-139.

Pellas, N. (2014). The influence of computer self-efficacy, metacognitive self-regulation and self-esteem on student engagement in online learning programs: Evidence from the virtual world of Second Life. *Computers in Human Behavior*, *35*(2), 157–170. http://doi.org/https://doi.org/10.1016/j.chb.2014.02.048

Pellas, N., & Peroutseas, E. (2016). Gaming in Second Life via Scratch4SL: Engaging High School Students in Programming Courses. *Journal of Educational Computing Research*, *54*(1), 108–143. http://doi.org/10.1177/0735633115612785

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

Ramalingham, V., & Wiedenbeck, S. (1998). Development and Validation of Scores on a Computer Programming Self-Efficacy Scale and Group Analyses of Novice Programmer Self-Efficacy. *Journal of Educational Computing Research*, *19*(4), 367–381.

Román-González, M. (2015). Computational Thinking Test : Design Guidelines and Content Validation Computational Thinking Test : Design Guidelines and Content Validation. *Proceedings of EDULEARN15 Conference*, (July), 2436–2444. http://doi.org/10.13140/RG.2.1.4203.4329

Saeli, M., Perrenet, J., Jochems, W. M. G., Zwaneveld, B., Nederland, O. U., & Centrum, R. D. M. (2011). Teaching Programming in Secondary School : A Pedagogical Content Knowledge Perspective. *Informatics in Education*, *10*(1), 73–88. http://doi.org/10.1145/2016911.2016943

Sayın, Z., & Seferoğlu, S. S. (2016). The impact of coding education and coding as a new 21st century skill on educational policies. *Academic Informatics Conference,* 3-5.

Soloway, E. (1993). Should we teach students to program? *Communications of the ACM*, *36*(10), 21–24. http://doi.org/10.1145/163430.164061

Şahiner, A., & Kert, S. (2016). Examination of the Studies Between 2006-2015 Related to the Concept of Computational Thinking. *European Journal of Science and Technology, European Journal of Science and Technology*, *5*, 38–43.

Werner, L., & Denning, J. (2009). Pair Programming in Middle School. *Journal of Research on Technology in Education*, *42*(1), 29–49. http://doi.org/10.1080/15391523.2009.10782540

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33. http://doi.org/10.1145/1118178.1118215

Yağcı, M. (2016). Effect of attitudes of information technologies (IT) preservice teachers and computer programming (CP) students toward programming on their perception regarding their self-sufficiency for programming. *International Journal of Human Sciences*, *13*(1). http://doi.org/10.14687/ijhs.v13i1.3502

Zell, E., & Krizan, Z. (2014). Do people have insight into their Abilities? A metasynthesis. *Perspectives on Psychological Science*, 9, 111-125.

Zimmerman, B. J., Bonner, S., & Kovach, R. (1996*). Developing Self-regulated learners: Beyond achievement to self-efficacy.* Washington, DC: American Psychological Association.

Ahi Evran Universitesi
Kırşehir Eğitim Fakültesi Dergisi

ISSN: 2147 - 1037

http://kefad.ahievran.edu.tr

Bilişim Öğretmeni Adaylarının Programlama İlişkin Öz-Yeterliklerinin Yordayıcısı Olarak Bilişimsel Düşünme ve Problem Çözmeye İlişkin Yansıtıcı Düşünme Becerileri

Serdar Çiftci,
Meltem Çengel,
Muhammed Paf

DOI:.....

Makale Bilgileri

Yükleme:25/08/2017 Düzeltme:01/12/2017 Kabul: 14/03/2018

Özet

Öz-yeterlik, bireyin kendi yeteneklerine ve kapasitesine olan inancıdır. Kişi amaca ulaşabileceğine inanıyorsa bu durum performansına olumlu yansımaktadır. Programlama becerisi, birçok beceri alanıyla ilişkili olup, öz-yeterlik kavramı programlama becerisinin gelişiminde önemli bir yer tutmaktadır. Programlamaya ilişkin öz-yeterliği ise kişinin program yazarken amaca ulaşabileceğine olan inancı anlamı çıkarılabilir. Bu bağlamda araştırmanın amacı, bilgisayar ve öğretim teknolojileri eğitimi bölümü öğretmen adaylarının programlamaya ilişkin öz-yeterlik algılarının, bazı demografik değişkenler, Bilişimsel düşünme ve problem çözmeye ilişkin yansıtıcı düşünme becerileri ile ne ölçüde yordandığını ortaya koymaktır. Çalışmada öğretmen adaylarından 4 bölümden oluşan anket yardımıyla veriler toplanmıştır. Ankette, demografik verileri içeren kişisel bilgiler bölümü, Ramalingam ve Wiedenbeck (1998) tarafından geliştirilen ve Altun ve Mazman (2012) tarafından Türkçe'ye uyarlanan Programlama İlişkin Öz yeterlik Ölçeği, Kızılkaya ve Aşkar (2009) tarafından geliştirilen Problem Çözmeye İlişkin Yansıtıcı Düşünme Ölçeği ve Korkmaz, Çakır ve Özden (2017) tarafından geliştirilen Bilgisayarca Düşünme Ölçeği kullanılmıştır.

Çalışmada kullanılan veri toplama aracı ile Türkiye'de bulunan dört farklı üniversitenin Bilgisayar ve Öğretim Teknolojileri Eğitimi bölümlerinde okuyan öğretmen adaylarından veriler elde edilmiştir. Verilerin analizinde Çoklu Regresyon analizi kullanılmıştır. Analiz sonuçlarına göre bağımsız değişkenler programlamaya ilişkin öz-yeterliği değeri ile anlamlı şekilde yordamaktadır ($F(3, 99)= 24.880$, $p< .001$). Modelin oluşmasına en önemli katkıyı bilişimsel düşünme, sonrasında problem çözmeye ilişkin yansıtıcı düşünme ve bilgisayar ile ilgili gelişmeleri takip etme katkı sunmaktadır. uyarlanmış R^2 değeri .413 düzeyindedir, ki bu durum model ile programlamaya ilişkin öz-yeterliğin %41'ini açıkladığını göstermektedir.

Anahtar Kelimeler: bilişim öğretmeni adayları, programlama ilişkin öz-yeterlik, bilişimsel düşünme, problem çözmeye ilişkin yansıtıcı düşünme

Sorumlu Yazar : Serdar Çiftci, Yrd.Doç.Dr., Adnan Menderes Üniversitesi, Türkiye, sciftci@gmail.com, https://orcid.org/0000-0001-5282-1861
Meltem Çengel, Yrd.Doç.Dr., Adnan Menderes Üniversitesi, Türkiye, meltemcengel@gmail.com, https://orcid.org/0000-0002-0255-4600
Muhammed Paf, Eğitim Programları ve Öğretim Yüksek Lisans Öğrencisi, muhammedpaf@gmail.com
Bu çalışma XV. European Conference on Social and Behavioral Sciences isimli konferansta sözlü olarak sunulmuştur.

Atıf için: *Bu bilgi KEFAD tarafından sağlanacaktır.*

Giriş

Son dönemde, bireylerin katılımlarını ve performanslarını etkileyen unsurlar arasında öz-yeterlik kavramı gibi bireysel ve içsel kaynaklar önem kazanmaktadır (Chen, 2017). Öz-yeterlik genel olarak bireyin görevi ne ölçüde yapacağına ilişkin inancıdır (Huffman, Whetten, & Huffman, 2013). Bandura'ya (1994) göre öz-yeterlik sadece performans düzeyi ile ilgili "kehanet"lerde bulunan bir kavram değildir. Öz-yeterlik güdülenmeye ilişkin bileşenleri uyarır ve görevin gerçekleştirilmesi için gerekli bilişsel beceriler ve çabayı tetikler (Kher, Downey, & Monk, 2013). Davidsson, Lazron ve Ljunggren (2013) öz-yeterliğe ilişkin inancın; belirlenen görevi başarmak için gerekli etkinlikleri seçme becerisi, çaba harcama, zorluklarla başa çıkma ve performans açısından etkili olduğunu belirtmektedirler.

Özellikle son yıllarda kazanan bilgisayar ve programlamaya ilişkin becerilerin öğrenilmesi önem kazanmaktadır (Bargury et al., 2012; Bers, Flannery, Kazakoff, & Sullivan, 2014; Grout & Houlden, 2014; Jones, Mitchell, & Humphreys, 2013). Bilgisayar ile ilgili becerilerin öğrenilmesinde öz-yeterlik kavramının son derece önemli olduğu bilinmektedir (Karsten, Mitra, & Schmidt, 2012). Ancak bu becerilerin öğrenilmesi sırasında, öz-yeterlik algılarının düşük olması nedeniyle, yani programlamayı baştan zor kabul etmeleri nedeniyle, programlama dersinde başarısız olmaları olası olduğuna değinilmektedir (Askar & Davenport, 2009). Igbaria ve Iivari (1995) sınıfta teknoloji kullanma ile ilgili olumsuz algının nedenlerinden birinin düşük teknoloji öz-yeterliği olabileceğine dikkati çekmektedir.

Ülkemizde programlamaya ilişkin öz-yeterlik ile ilgili çalışmalar incelendiğinde, programla dersi alan öğrencilerin özdüzenleyici stratejileri ile başarıları arasındaki ilişkilere odaklanan (Haşlaman & Aşkar, 2007), Bilgisayar ve Öğretim Teknolojileri Eğitimi (BÖTE) ve farklı bölümlere devam eden ve programlama dersi alan öğrencilerin programlamaya ilişkin öz-yeterlik algıları (Gezgin & Adnan, 2016; Mazman & Altun, 2013), programlamaya ilişkin öz-yeterlik ve tutum arasındaki ilişkiler (Özyurt & Özyurt, 2015; Yağcı, 2016) gibi konular ele alınmıştır. Bununla birlikte, Psycharis ve Kallia (2017) bilgisayar programlama becerisinin, lise öğrencilerinin sorgulama becerileri, problem çözme ve matematik öz-yeterlikleri üzerindeki etkisini incelemiştir. Başka bir araştırmada ise; Bilgisayar Bilimleri öğrencilerine verilen problem çözme öğretiminin programlama öz-yeterliği ve başarısı üzerindeki etkileri incelenmiştir (Maddrey, 2011).

Var olan çalışmalara gözden geçirilerek; bu çalışmanın temel amacı, programlamaya ilişkin öz-yeterliğin yordayıcılarını ortaya koymak olarak belirlenmiştir. Bu amaçla, programlamaya ilişkin öz-yeterlik ile ilişkili olduğu düşünülebilen, problem çözmeye ilişkin yansıtıcı düşünme, bilişimsel düşünme ve bazı demografik değişkenler ele alınmıştır.

Kuramsal Çerçeve

Problem Çözmeye ilişkin Yansıtıcı Düşünme

Program yazma; problem çözme, soyutlama, matematiksel mantık, sınama, hata ayıklama ve sorunları çözme gibi karmaşık ve zorlu görevleri içermektedir (Saeli ve diğerleri, 2011). Program yazma süreci, problem çözme sürecine benzer aşamalar içerir. Soloway (1993)'e göre; programlama süresince, bireyler problem çözme/tasarlama/düşünme stratejilerini etkin olarak kullanırlar. Bunun temel nedeni, programlama esnasında, öğrenenlerin öncelikle problemin nasıl çözüleceğini matematiksel olarak bulmaları, sonrasında bunu bilgisayara nasıl aktaracakları ve doğru olarak yazılım diline aktarmayı düşünmelerinin gerekliliğidir (Psycharis & Kallia, 2017). Program yazma süreci, problemin nasıl etkili çözülebileceğine ilişkin derinlemesine okuma ve bilişsel farkındalık becerilerini de gerektirir. Bu nedenle, program yazma, programın niteliğini geliştirmek üzere kararların ve eylemlerin sürekli gözden geçirilmesini ve alternatiflerin üretilmesini içerir (Kalelioğlu, 2015). Govender ve diğerlerine (2014) göre ise program yazma sürecinde gerekli olan bazı beceriler açık olarak öğretilmemektedir. Bu noktada gerekli olan becerilerden biri olarak problem çözmenin öğretiminin, yalnızca sorunlar üzerinde uygulama yapmak değil, aynı zamanda problem çözme sürecinin öğretilmesini içermesi gerektiği belirtilmektedir.

Özellikle problemin çok iyi yapılandırılmadığı örneklerde (ill-structured problems); süreç belirsizleşmekte, pek çok çözüm ve değerlendirme olasılığı bulunmaktadır. Bu tür problemlerin çözümünde sürece ilişkin olarak yansıtıcı düşünme; sürecin kontrolü, izlenmesi ve deneyim kazanılması aşamalarını içermesi nedeniyle önemlidir (Lee, Teo, & Zealand, 2011). Yansıtıcı düşünme, eleştirel düşünme sürecinin önemli bir parçasıdır. Ayrıca analiz etme süreçleri ve neler olduğuna ilişkin karar süreçlerini içermektedir. Öğrencilere bir adım geri atıp, bir sorunu nasıl çözdükleri ile ilgili fikir sağladığı ve hedefledikleri çözüme ulaşabilmek için kullandıkları stratejilerin ne ölçüde işe yaradığını anlamalarını sağladığı için de önemlidir (Kalelioğlu, 2015). Yansıtıcı düşünme, durumdan kuşkulanma, tereddüt etme, şaşırma, zihinsel güçlük içerme, sorgulama, şüpheyi giderecek materyal bulma etkinliklerini kapsaması bakımından düşünme adına uygulanılan işlemlerden ayrılmaktadır. Yansıtma kavramının farklı bağlamlarda farklı tanımlarla kullanılmaktadır. Bu çalışmada yansıtıcı düşünme problem çözme bağlamı ile sınırlandırılmıştır.

Programlama süreçleri, problem çözme ve yansıtıcı düşünme arasındaki ilişkilere odaklanan pek çok çalışma bulunmaktadır (Fessakis, Gouli, & Mavroudi, 2013; Liao & Bright, 1991). Örneğin; Bergin, Reilly ve Traynor (2005), bilişüstü ve yönetimsel süreçlere ilişkin olarak daha çok strateji kullanan öğrencilerin programlama süreçlerinde daha başarılı olduğunu belirtmektedirler. Akcaoglu ve Koehler (2014) ise oyun-tasarımı ve öğrenme grubuna katılan öğrencilerin problem çözme

becerilerinin katılmayanlara göre anlamlı şekilde arttığını belirtmektedirler. Ke (2014) ise ele aldığı örnek olayda, bilgisayarda oyun programlamanın; öğrencilerin matematik ile ilişkili kavram ve problem çözmeye ilişkin becerilerine katkı sunduğunu belirtmektedir. Psycharis ve Kallia (2017)'nın çalışması ise programlamanın öğrencilerin sorgulama, problem çözme ve matematiğe ilişkin öz-yeterlikleri üzerinde anlamlı bir etkisi olup olmadığına ilişkindir. Çalışmaya göre, bilgisayarda programlama, öz-yeterlik ile ilgili belirleyiciler üzerinde anlamlı bir etki yaparken, problem çözme üzerinde anlamlı bir etkide bulunmamaktadır.

Bilişimsel Düşünme

Bilişimsel düşünme (Computational Thinking) ile ilgili alan yazında isimlendirme karmaşası söz konusudur. Türkçe alanyazında yapılan çalışmalarda farklı isimlendirmeler bulunmaktadır. Komputasyonel düşünme (Şahiner & Kert, 2016), bilgi-işlemsel düşünme (Demir & Seferoglu, 2017), bilgisayarca düşünme (Çatlak, Tekdal & Baz, 2015), bilişimsel düşünme (Sayın & Seferoğlu, 2016), hesaplamalı düşünme (MEB, 2017), kompütasyonel düşünme (Aldağ & Tekdal, 2015; Şahiner &Kert, 2016) ve bilgi-işlemsel düşünme (Barut, Tuğtekin & Kuzu, 2016; Kalelioğlu, Gülbahar, Akçay ve Dogan, 2015) farklı kullanılan isimlendirmelerden bazılarıdır. Buna ek olarak; programlama yerine yazılım geliştirme, kodlama, program yazma, bilgisayar programı yazma gibi ifadeler de kullanılmaktadır. Bu çalışmada "bilişimsel düşünme" ve "programlama" ifadeleri temel alınmıştır.

Bilgisayarda program yazma becerileri ile ilişkilendirilen önemli değişkenlerden biri de bilişimsel düşünmedir (Román-González, 2015). İlk olarak Wing (2006) tarafından ortaya atılan bilişimsel düşünme kavramı, problem çözme yollarını, sistemler tasarlamayı ve insan davranışlarını anlamayı içerir. Bilgisayara ilişkin kavramları bilmek, "değişken" gibi programlama sürecinde kullanılan kavramları bilmeye işaret ederken; uygulamaları bilmek programlama sürecinde ortaya çıkan, döngü, tekrar, koşul gibi problem çözme uygulamalarını anlamayı; bakış açılarını anlamak ise bireyin kendisinin ve diğerlerinin teknolojik dünya ile ilişkisini kavramı içerir (Lye & Koh, 2014). Bilişimsel düşünme yalnızca bilgisayar bilimcilerin kullanması gereken bir kavram değildir (Kim, Kim, & Kim, 2013). Hatta Kafai ve Burke (2013), bilişimsel düşünme kavramından bahsedebilmek için teknolojik araçlara gereksinim duyulmadığına değinmektedirler.

Wing (2006) bilişimsel düşünmeyi, eleştirel düşünme ve var olan bilgilerin matematik, fen ve genel olarak FETEMM (Fen bilimleri, teknoloji, mühendislik ve matematik-STEM) alanındaki karmaşık problemlerin çözümüne uygulanması olarak açıklamaktadır. Aho (2012)'ya göre bilişimsel düşünme problemlerin parçalara ayrılması, soyutlama, mantıksal düşünme, algoritma yaratma ve hata ayıklama olarak formüle edilebilir. Son dönemde ise bilişimsel düşünme; farklı hiyerarşik seviyelerde soyutlama, algoritmik düşünme, otomatizasyon, parçalara ayırma, modelleme, kalıplar

çıkarma, tekrarlama, ölçekleme ve sembolik temsil gibi problem çözme stratejilerini ve bilgisayar bilimi yapılarını kapsayan bir kavram olarak tanımlanabilir (Grover & Pea, 2013; Korkmaz, Çakir, & Özden, 2017; Ni & Guzdial, 2012; Psycharis & Kallia, 2017; Werner & Denning, 2009). Bununla birlikte bu kavramı, analitik ve problem çözme becerilerini içermenin yanında, alışkanlıklar, hata ve sorunlar ile karşılaşıldığında takınılan tutum ve bilgisayar bilimlerinde kullanılan yaklaşımları benimseme ile ilişkilendiren çalışmalar da bulunmaktadır (Bers et al., 2014).

Doğrudan bilgisayarda programlamaya ilişkin öz-yeterlik ve bilişimsel düşünme arasındaki ilişkilere odaklanan çalışmalar olmamakla birlikte, Pellas (2014) çalışmasında öğrencilerin derse katılımının, bilgisayara ilişkin öz-yeterlik, üst bilişsel özdüzenleyici stratejiler ve benlik saygısı tarafından anlamlı şekilde yordandığını ve bu bağımsız değişkenler arasında anlamlı düzeyde ilişki olduğunu belirtmektedir. Robotik ile programlamaya ilişkin yürütülen deneysel çalışmada Jaipal-Jamani ve Angeli (2017), öğretmen adaylarının robotik ile öğretime ilişkin öz-yeterlik inançlarının, alan bilgisi düzeylerinin ve bilişimsel düşünme düzeylerinin arttığına işaret etmektedir. Başka bir çalışmada ise (Psycharis & Kallia, 2017), öğrencilere matematik ile ilişkilendirilerek gerçekleştirilen programlama öğretiminin matematiğe ilişkin öz-yeterlikleri üzerinde etkisi olduğuna değinmektedir. Pellas ve Peroutseas (2016) nitel yöntem uyguladıkları çalışmalarında, programla dersin yürütülen çalışmalarında, öğrencilerin programlamaya ilişkin yeterlikleri ve bilişimsel düşünmeleri arasında ilişkilere işaret ettiğine değinmektedir.

Yöntem

Katılımcılar

Araştırmanın 166 katılımcısı bulunmaktadır. Bunların 93'ü erkek (%56), 73'ü kadındır (%44). Katılımcılar Adnan Menderes Üniversitesi (n=48, %28.9), Muğla Sıtkı Koçman Üniversitesi (n=14, %8.4), Süleyman Demirel Üniversitesi (n=53, %31.9) ve Konya Selçuk Üniversitesi, Ereğli Eğitim Fakültesi'nden (n=50, %30.1) olmak üzere dört farklı üniversiteye devam etmektedir. Araştırma katılımcılarının %48.2 (n=80) ikinci sınıfa devam ederken, %22.9 (n=38) üçüncü sınıfa, %28.9 (n=48) dördüncü sınıfa devam etmektedir. %24.1 (n=40) Anadolu Lisesi, %65.7 (n=109) Mesleki veya teknik lise, %10.2'si (n=17) ise diğer lise türlerinden mezun olmuştur. 160 katılımcının (%96.4) bilgisayarı var iken, 6 katılımcının (%3.6) bilgisayarı yoktur. 17 öğrenci (%14.8) bilgisayar ile ilgili gelişmeleri takip ederken, 98 öğrenci (n=59.0) bu gelişmeleri takip etmemektedir.

Araçlar

Uygulanan araçta öğrencilerin bazı demografik ve bilgisayar kullanma alışkanlıklarını belirlemeye yönelik 10 maddelik kişisel bilgiler formu, programlamaya ilişkin öz-yeterlik ölçeği, problem çözmeye ilişkin yansıtıcı düşünme ölçeği ve bilişimsel düşünme ölçeği kullanılmıştır.

Üniversite öğrencilerinin C++ programını kullanabilme öz-yeterliklerini belirmek amacıyla Ramalingam ve Wiedenbeck (1998) tarafından geliştirilen Programlamaya İlişkin Öz-yeterlik Ölçeği'nin Türkçeye uyarlamasını Altun ve Mazman (2012) tarafından gerçekleştirilmiştir. Uyarlama çalışması sonucunda dokuz maddeden oluşan iki faktörlü bir yapıya ulaşılmıştır. "Karmaşık programlama görevleri" ve "basit programlama görevleri" olmak üzere iki boyuttan oluşan ölçeğin iç tutarlılık katsayısı .928 bulunmuş ve bu yapı toplam varyansın %80,814'ünü açıklamıştır. Bu yapıya ilişkin model doğrulayıcı faktör analizi çalışması da gerçekleştirilmiştir.

Kızılkaya ve Aşkar (2009) tarafından geliştirilen Problem Çözmeye İlişkin Yansıtıcı Düşünme ölçeği 14 madde ve üç alt boyuttan oluşmaktadır. Temel alınan üç boyut; sorgulama, değerlendirme ve nedenleme boyutlarıdır. Sorgulama boyutuna örnek olarak "Problemi okuduğumda verilen ve istenenleri belirlemek için kendime sorular sorarım" maddesi verilebilir. Değerlendirme boyutuna örnek olarak "Çözüm yollarımı tekrar tekrar değerlendirip bir sonraki problemi daha iyi çözmeye çalışırım" maddesi, nedenleme boyutuna örnek olarak "Problem çözerken, yaptığım işlemlerin nedenini düşünerek, bulduğum sonuçla ilişkisini kurmaya çalışırım" gösterilebilir. Ölçek maddeleri 5'li likert tipine göre puanlanmıştır. Maddelerin içerdiği eylem sıklıkları "Her zaman", "Çoğu zaman", "Bazen", "Nadiren", "Hiçbir zaman" seviyelerinde düzenlenmiştir. Bu seviyeler; Her zaman=5, Çoğu zaman=4, Bazen=3, Nadiren=2, Hiçbir zaman=1 olarak puanlanmıştır. Ölçeğe ilişkin doğrulayıcı faktör analizi sonuçları GFI= 0,92, AGFI= 0,89, NNFI= 0,93, CFI= 0,95, RMSR= 0,08, RMSEA= 0.071 olarak hesaplanmıştır.

Korkmaz, Çakır ve Özden (2017) tarafından geliştirilen ve üniversite öğrencileri üzerinde uygulanan ölçek toplam 22 maddeden ve beş faktörden oluşmaktadır. Ölçek beşli Likert tipli bir ölçek olup, beş faktör altında toplanabilen 22 maddeden oluşmaktadır. Ölçeğin faktör yapılarının doğrulanmasına yönelik doğrulayıcı faktör analizi yapılmıştır. Ölçek yaratıcılık (4 madde), algoritmik düşünme (4 madde), işbirliklilik (4 madde), eleştirel düşünme (4 madde), problem çözme (6 madde) boyutlarından oluşmaktadır. Ölçeğin toplam cronbach alpha güvenirlik puanı, .81 düzeyindedir. Ölçek toplam varyansın %56.1'ini açıklamaktadır.

Analiz

Yaklaşık 20 dakika süren uygulamalar sonrasında veriler SPSS ortamına aktarılmış ve verilerin analizi için çoklu doğrusal regresyon analizi yapılmıştır. Çoklu regresyon analizinin gerçekleştirilebilmesi için karşılanması gereken varsayımlar, yordayan değişkenlerin ve bağımlı değişkenin doğrusal olması, hataların ve artıkların normal olarak dağılması ve yordayan ile ilişkisiz olmasıdır. Bununla birlikte, çoğunlukla sorun oluşturan durumlardan biri, çoklu bağlaşım sorunudur. Çoklu bağlaşım, değişkenler arasında yüksek ilişki olduğu durumlarda gözlenmektedir. Çoklu

bağlaşımın olmamasına, tolerans ve VİF değerleri kontrol edilerek karar verilmiştir (Leech, Barrett, & Morgan, 2005).

Bulgular ve Yorum

Programlamaya ilişkin öz-yeterliğin, problem çözmeye ilişkin yansıtıcı düşünme, bilişimsel düşünme ve bilgisayar ile ilişkili yayınları takip etmeye göre yordanıp yordanamayacağını belirlemek üzere çoklu doğrusal regresyon analizi yapılmıştır.

Öncelikle Tablo 1'de ilgili değişkenlere ilişkin betimsel ve ilişkisel istatistikler sunulmuştur.

Tablo 1. *Programlamaya İlişkin Öz-yeterlik ve Bağımsız değişkenlerin Ortalama, Standart Sapma ve İlişki Düzeyleri*

Değişkenler	X	SD	1	2	3
Programlama İlişkin Öz-yeterlik	41.796	12.371	.563*	.554*	-.365*
1. Bilişimsel Düşünme	84.796	11.310	-	.585*	-.233**
2. Problem çözmeye ilişkin yansıtıcı düşünme	54.815	9.295		-	-.293*
3. Bilgisayar ile ilgili gelişmeleri takip etme	.845	.364			-

* p< .001

** p< .009

Tablo 1'e göre, programlamaya ilişkin öz-yeterlik ile bilişimsel düşünme arasında .563 düzeyinde, problem çözmeye ilişkin yansıtıcı düşünme arasında, .554 düzeyinde ve bilgisayar ile ilgili gelişmeleri takip etme arasında -.365 düzeyinde anlamlı ilişkiler bulunmaktadır (p< .001). bilişimsel düşünme ile problem çözmeye ilişkin yansıtıcı düşünme arasında .585 düzeyinde, bilgisayar ile ilgili gelişmeleri takip etme arasında -.233 düzeyinde anlamlı ilişkiler, problem çözmeye ilişkin yansıtıcı düşünme ile bilgisayar ile ilgili gelişmeleri takip etme arasında -.293 düzeyinde anlamlı ilişkiler bulunmaktadır. Tablo 2'de ilgili değişkenleri içeren regresyon modeli özeti sunulmuştur.

Tablo 2. *Programlama İlişkin Öz-yeterliğin bilişimsel düşünme, problem çözmeye ilişkin yansıtıcı düşünme ve bilgisayar ile ilgili gelişmeleri takip etme ile yordanmasına ilişkin Çoklu Regresyon Model Özeti*

Değişken	B	Std. Hata	β
Bilişimsel Düşünme	.377	8.974	.345
Problem çözmeye ilişkin yansıtıcı düşünme	.391	.103	.294
Bilgisayar ile ilgili gelişmeleri takip etme	-6,756	.127	.199
Sabit	-3,856	2.706	

Analiz sonuçlarına göre bağımsız değişkenler programlamaya ilişkin öz-yeterliği değeri ile anlamlı şekilde yordamaktadır (F (3, 99)= 24.880, $p< .001$). Üç bağımsız değişken de modelin oluşmasına anlamlı katkı sunmaktadır. Tablo 2'de sunulan beta değerlerine göre, modelin oluşmasına en önemli katkıyı bilişimsel düşünme, sonrasında problem çözmeye ilişkin yansıtıcı düşünme ve bilgisayar ile ilgili gelişmeleri takip etme katkı sunmaktadır. Analiz sonuçlarına göre uyarlanmış R^2 değeri .413 düzeyindedir, ki bu durum model ile programlamaya ilişkin öz-yeterliğin %41'ini açıkladığını göstermektedir.

Tartışma ve Öneriler

Öz-yeterlik genel olarak bireyin bir beceriyi ne ölçüde yapacağına ilişkin inancı (Huffman, Whetten, & Huffman, 2013), programlamaya ilişkin öz yeterlik ise bireyin program yazarken amaca ulaşabileceğine olan inancı olarak etmektedir. Askar ve Davenport (2009) programlama becerilerin öğrenilme sürecinde, öz-yeterlik algılarının düşük olmasının programlama dersinde başarısız olma durumlarını tetikleyebildiğini ifade etmektedirler.Araştırma kapsamında yapılan analiz sonuçlarına göre; programlamaya ilişkin öz-yeterliğin bilişimsel düşünme, problem çözmeye ilişkin yansıtıcı düşünme ve bilgisayar ile ilgili gelişmeleri takip etme tarafından anlamlı şekilde yordandığı göstermektedir. Jaipal-Jamani ve Angeli (2017) ilkokul öğretmen adaylarının robotik öz-yeterliği ve bilişimsel düşünmelerine ilişkin yürüttükleri deneysel çalışmada, öğrencilerin robotiğe ilişkin öz-yeterlikleri ve bilişimsel düşünme becerilerinin deneysel süreçten anlamlı düzeyde etkilendiğini belirlemişlerdir.

Bilişimsel düşünme ve problem çözme becerisi ilişkili olmakla birlikte, bilişimsel düşünme ile birlikte gelişen problem çözme becerisi matematik gibi diğer alanlara aktarılabilmektedir. Wing (2006) tanımında BD'yi problem çözmenin bir yolu olarak ifade etmektedir. Problem çözmeye ilişkin beceriler ile programlamaya ilişkin öz-yeterlik arasında sıkı bağlar olduğu Maddrey (2011)'in çalışmasında da ortaya konulmuştur. Bu çalışmada matematik içermeyen problem çözümü

öğretiminin öğrencilerin programlamaya ilişkin öz-yeterliğini arttığını gözlenmiştir. Programlama öğretiminin sorgulama becerileri, problem çözme ve matematik öz-yeterliği arasındaki ilişkileri araştıran başka bir araştırmada ise programlama öğretiminin sorgulama becerilerini ve matematik öz-yeterliğini anlamlı düzeyde etkilediği, ancak problem çözme becerilerine anlamlı düzeyde katkı sunduğu belirtilmektedir (Psycharis & Kallia, 2017). Bununla birlikte, Kalelioğlu (2015) ilköğretim öğrencilerine programlama öğretimini içeren deneysel çalışmasında, programlama becerisi öğretiminin problem çözmeye ilişkin yansıtıcı düşünme üzerinden anlamlı bir etkisi olmadığını ifade etmiştir.

Araştırma sonuçlarında dikkat çeken bir diğer bulgu ise programlamaya ilişkin öz-yeterlik ile bilgisayar ile ilgili gelişmeleri takip etme arasında ters yönlü bir ilişki vardır. Bu sonuca göre programlamaya ilişkin öz-yeterlik becerileri yüksek olan bireylerin alandaki gelişmeleri daha az takip ettikleri söylenilebilir. Bu durum kendini-değerlendirme türündeki ölçeklerde orta çıkabilecek yanlılık kavramı ile ilişkili olabilir. Örneğin, bilgisayar öz yeterliliği kavramı ile ilgili bazı çalışmalar, özyeterliliğin önemli olduğu kadar, kendini-değerlendirmeye dayalı bu kavramın yansızlıktan uzak olmasının da önemli olduğunu ifade etmektedirler. Yansızlık kendi performansını olduğundan fazla ya da az algılamama ile ilişkilidir (Pajares & Graham, 1999). Genel olarak, kendi performansını ve özyeterliliğini olduğundan biraz daha iyi algılamak sonrasındaki çabayı ve bu çabanın sürekliliğini arttırabilir (Bandura, 1986; Pajares, 1997). Ancak kendi-performansını olanın çok üzerinde algılayan bireyler çalışma yöntemini değiştirmek veya kendini daha çok geliştirme ile ilgili çaba harcamayabilir (Zimmerman, Bonner, and Kovach, 1996). Konu ile daha çok ilgili öğrenciler, kendi performanslarını daha doğru değerlendirebilmektedirler (Zell ve Krizan, 2014). Benzer şekilde, Ehrlinger ve diğerleri (2008) de belirli bir alanda, herhangi bir konu ile ilgili daha deneyimli kişilerin kendi performanslarına ve bilişsel kapasitelerine ilişkin daha çok geri bildirim aldıklarına işaret etmektedirler.

Öğrencilerin bilgisayara ilişkin özyeterlik bildirimlerindeki doğruluk ve yanlılığı inceleyen çalışmada, daha deneyimli öğrencilerin özyeterliklerini daha düşük değerlendirdikleri gözlenmiştir. Bununla birlikte, bilişsel yeteneği daha yüksek öğrenciler kendini-değerlendirme ölçeklerinde performansları ile daha tutarlı ve doğru bildirilerde bulunmuşlardır (Aesaert, Voogt, Kuiper, van Braak, 2017). Buradan yola çıkarak gelişmeleri daha çok takip eden öğrencilerin kendi yeterliliklerini daha düşük buldukları düşünülebilir.

Araştırma bulgularından yola çıkarak, farklı kademelerde uygulamaya konan programlama derslerinin bir parçası olarak problem çözmeye ilişkin yansıtıcı değerlendirme ve bilişimsel düşünme becerilerinin eğitim progrmalarına dahil edilmesi önerilebilir. Bununla birlikte, programlamaya ilişkin özyeterliği yordayan diğer değişkenlerin belirlenmesine yönelik çalışmalar yürütülebilir. Diğer bir

araştırma konusu olarak, programlamaya ilişkin gerçek performans, programlamaya ilişkin öz-yeterlik ve bu iki kavram arasındaki ilişkileri etkileyen diğer değişkenleri inceleyen çalışmalara gereksinim duyulduğu söylenebilir.

Kaynakça

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, *55*(7), 833–835. http://doi.org/10.1093/comjnl/bxs074

Akcaoglu, M., & Koehler, M. J. (2014). Cognitive outcomes from the Game-Design and Learning (GDL) after-school program. *Computers and Education*, *75*, 72–81. http://doi.org/10.1016/j.compedu.2014.02.003

Aldağ, H., & Tekdal, M. (2015). Bilgisayar kullanımı ve programlama öğretiminde cinsiyet farklılıkları. *Proceeding of 1. Uluslararası Çukurova Kadın Çalışmaları Kongresi*, 236–243.

Altun, A., & Mazman, S. G. (2012). Programlamaya ilişkin öz yeterlilik algısı ölçeğinin Türkçe formumun geçerlilik ve güvenirlik çalışması. *Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi*, *3*(2), 297–308.

Askar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java programming among engineering students. *The Turkish Online Journal of Educational Technology*, *8*(1), 1303–6521. Retrieved from http://files.eric.ed.gov/fulltext/ED503900.pdf

Bandura, A. (1986). *Social foundations of thought and action: A social cognitive theory.* Englewood Cliffs, NJ: Prentice Hall.

Bandura, A. (1994). Self-efficacy. In V. S. Ramachaudran (Ed.), Encyclopedia of human behavior (pp. 71–81). New York: Academic.

Bargury, I. Zur, Haberman, B., Cohen, A., Muller, O., Zohar, D., Levy, D., & Hotoveli, R. (2012). Implementing a new Computer Science Curriculum for middle school in Israel. *2012 Frontiers in Education Conference Proceedings*, 1–6. http://doi.org/10.1109/FIE.2012.6462365

Barut, E., Tuğtekin, U., & Kuzu, A. (2016). Robot uygulamalar ile bilgi işlemsel düşünme becerilerine bakış. Içinde *3rd International Conference on New Trend in Education (ICNTE 2016)*.

Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. *First International Workshop on Computing Education Research*, 81–86. http://doi.org/10.1145/1089786.1089794

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, *72*, 145–157. http://doi.org/10.1016/j.compedu.2013.10.020

Chen, I. S. (2017). Computer self-efficacy, learning performance, and the mediating role of learning engagement. *Computers in Human Behavior*, *72*, 362–370. http://doi.org/10.1016/j.chb.2017.02.059

Çatlak, Ş., Tekdal, M., & Baz, F. Ç. (2015). Scratch Yazılımı İle Programlama Öğretiminin Durumu: Bir Doküman İnceleme Çalışması. *Journal of Instructional Technologies & Teacher Education*, *4*(3). Tarihinde adresinden erişildi http://www.jitte.org/article/view/5000163313

Davidsson, K., Larzon, L., & Ljunggren, K. (2013). Self-Efficacy in Programming among STS Students. Retrieved from http://www.it.uu.se/edu/course/homepage/datadidaktik/ht10/reports/Self-Efficacy.pdf

Demir, Ö., & Seferoglu, S. S. (2017). Yeni Kavramlar, Farklı Kullanımlar: Bilgi-İşlemsel Düşünmeyle İlgili Bir Değerlendirme (ss. 801–830).

Ehrlinger, J., Johnson, K., Banner, M., Dunning, D., & Kruger, J. (2008). Why the unskilled are unaware: Further explorations of (absent) self-insight among the incompetent. Organizational Behavior and Human Decision Processes, 105,98-121.

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

Gezgin, D. M., & Adnan, M. (2016). Makine Mühendisliği ve Ekonometri Öğrencilerinin Programlamaya İlişkin Öz Yeterlik Algılarının İncelenmesi. *Ahi Evran Üniversitesi Kırşehir Eğitim Fakültesi Dergisi*, *17*(2), 509–525.

Govender, I., Govender, D., Havenga, M., Mentz, E., Breed, B., Dignum, F., & Dignum, V. (2014). Increasing self-efficacy in learning to program: exploring the benefits of explicit instruction for problem solving. *TD The Journal for Transdisciplinary Research in Southern Africa*, *10*(1), 187–200.

Grout, V., & Houlden, N. (2014). Taking Computer Science and Programming into Schools: The Glyndŵr/BCS Turing Project. *Procedia - Social and Behavioral Sciences*, *141*, 680–685. http://doi.org/10.1016/j.sbspro.2014.05.119

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

Haşlaman, T., & Aşkar, P. (2007). Programlama Dersi ile İlgili Özdüzenleyici Öğrenme Stratejileri ve Başarı Arasındaki İlişkinin İncelenmesi. *Hacettepe Üniversitesi Eğitim Fakültesi Dergisi*, *32*, 110–122.

Huffman, A. H., Whetten, J., & Huffman, W. H. (2013). Using technology in higher education: The influence of gender roles on technology self-efficacy. *Computers in Human Behavior*, *29*(4), 1779–1786. http://doi.org/10.1016/j.chb.2013.02.012

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

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of Robotics on Elementary Preservice Teachers' Self-Efficacy, Science Learning, and Computational Thinking. *Journal of Science Education and Technology*, *26*(2), 175–192. http://doi.org/10.1007/s10956-016-9663-z

Jones, S. P., Mitchell, B., & Humphreys, S. (2013). Computing at school in the UK : from guerrilla to gorilla. *Communications of the ACM*, (April), 1–13. http://doi.org/10.1016/j.compedu.2013.10.020

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

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, *52*, 200–210. http://doi.org/10.1016/j.chb.2015.05.047

Kalelioğlu F, Gülbahar Y, Akçay S, Dogan D. (2014). Curriculum Integration Ideas for Improving the Computational Thinking Skills of Learners through Programming via Scratch.. *7 th International Conference on Informatics in Schools: Situation, Evolution and Perspectives*: İstanbul; 22/09/2014 - 25/09/2014

Karsten, R., Mitra, A., & Schmidt, D. (2012). Computer self-efficacy : A meta-analysis. *Journal of Organizational Ans End User Computing*, *24*(4), 54–80.

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers and Education*, *73*, 26–39. http://doi.org/10.1016/j.compedu.2013.12.010

Kher, H. V., Downey, J. P., & Monk, E. (2013). A longitudinal examination of computer self-efficacy change trajectories during training. *Computers in Human Behavior*, *29*(4), 1816–1824. http://doi.org/10.1016/j.chb.2013.02.022

Kim, B., Kim, T., & Kim, J. (2013). Paper-and-Pencil Programming Strategy toward Computational Thinking for Non-Majors: Design Your Solution. *Journal of Educational Computing Research*, *49*(4), 437–459. http://doi.org/10.2190/EC.49.4.b

Kizilkaya, G., & Aşkar, P. (2009). Problem çözmeye yönelik yansitici düşünme becerisi ölçeğinin Geliştirilmesi. *Egitim ve Bilim*, *34*(154), 82–92.

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, *72*, 558–569. http://doi.org/10.1016/j.chb.2017.01.005

Lee, C., Teo, T., & Zealand, N. (2011). Shifting Pre-service Teachers â€™ Metaeognition Through Problem Solving, *3*, 570–578.

Leech, N. L., Barrett, K. C., & Morgan, G. A. (2005). *SPSS for Intermediate Statistics: Use and Interpretation* (2nd ed.). New Jersey: Lawrence Erlbaum Associates.

Liao, Y.-K. C., & Bright, G. W. (1991). Effects of Computer Programming on Cognitive Outcomes: A Meta-Analysis. *Journal of Educational Computing Research*, *7*(3), 251–268. http://doi.org/10.2190/E53G-HH8K-AJRR-K69M

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through

programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. http://doi.org/10.1016/j.chb.2014.09.012

Maddrey, E. (2011). *The Effect of Problem-Solving Instruction on the Programming Self- efficacy and Achievement of Introductory Computer Science Students*. *Information Sciences*. Nova Southeastern University.

Mazman, S. G., & Altun, A. (2013). The effect of introductory to programming course on programming self efficacy of CEIT students. *Journal of Instructional Technologies & Teacher Education*, *2*(3), 24–29.

Ni, L., & Guzdial, M. (2012). Who AM I?: Understanding high school computer science teachers' professional identity. *SIGCSE '12: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 499–504. http://doi.org/10.1145/2157136.2157283

Özyurt, Ö., & Özyurt, H. (2015). A study for determinin computer programming students' attitudes towards progmming and their programming self-efficacy. *Eğitimde Kuram ve Uygulama Articles /Makaleler Journal of Theory and Practice in Education*, *11*(1), 51–67.

Pajares, F., & Graham, L. (1999). Self-efficacy, motivation constructs, and mathematics performance of entering middle school students. *Contemporary Educational Psychology*, 24, 124-139.

Pellas, N. (2014). The influence of computer self-efficacy, metacognitive self-regulation and self-esteem on student engagement in online learning programs: Evidence from the virtual world of Second Life. *Computers in Human Behavior*, *35*(2), 157–170. http://doi.org/https://doi.org/10.1016/j.chb.2014.02.048

Pellas, N., & Peroutseas, E. (2016). Gaming in Second Life via Scratch4SL: Engaging High School Students in Programming Courses. *Journal of Educational Computing Research*, *54*(1), 108–143. http://doi.org/10.1177/0735633115612785

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

Ramalingham, V., & Wiedenbeck, S. (1998). Development and Validation of Scores on a Computer Programming Self-Efficacy Scale and Group Analyses of Novice Programmer Self-Efficacy. *Journal of Educational Computing Research*, *19*(4), 367–381.

Román-González, M. (2015). Computational Thinking Test : Design Guidelines and Content Validation Computational Thinking Test : Design Guidelines and Content Validation. *Proceedings of EDULEARN15 Conference*, (July), 2436–2444. http://doi.org/10.13140/RG.2.1.4203.4329

Saeli, M., Perrenet, J., Jochems, W. M. G., Zwaneveld, B., Nederland, O. U., & Centrum, R. D. M. (2011). Teaching Programming in Secondary School : A Pedagogical Content Knowledge Perspective. *Informatics in Education*, *10*(1), 73–88. http://doi.org/10.1145/2016911.2016943

Sayın, Z., & Seferoğlu, S. S. (2016). Yeni bir 21. yüzyıl becerisi olarak kodlama eğitimi ve kodlamanın eğitim politikalarına etkisi. *Akademik Bilişim Konferansı*, 3–5.

Soloway, E. (1993). Should we teach students to program? *Communications of the ACM*, *36*(10), 21–24. http://doi.org/10.1145/163430.164061

Şahiner, A., & Kert, S. (2016). Komputasyonel Düşünme Kavramı ile İlgili 2006-2015 Yılları Arasındaki Çalışmaların İncelenmesi. *European Journal of Science and Technology, Avrupa Bilim ve Teknoloji Dergisi*, *5*, 38–43.

Werner, L., & Denning, J. (2009). Pair Programming in Middle School. *Journal of Research on Technology in Education*, *42*(1), 29–49. http://doi.org/10.1080/15391523.2009.10782540

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33. http://doi.org/10.1145/1118178.1118215

Yağcı, M. (2016). Effect of attitudes of information technologies (IT) preservice teachers and computer programming (CP) students toward programming on their perception regarding their self-sufficiency for programming Bilişim teknolojileri (BT) öğretmen adaylarının. *International Journal of Human Sciences*, *13*(1). http://doi.org/10.14687/ijhs.v13i1.3502

Zell, E., & Krizan, Z. (2014). Do people have insight into their Abilities? A metasynthesis. *Perspectives on Psychological Science*, 9, 111-125.

Zimmerman, B. J., Bonner, S., & Kovach, R. (1996*). Developing Self-regulated learners: Beyond achievement to self-efficacy*. Washington, DC: American Psychological Association.