Examination of the Transitions between Modal Representations in Coding

Mustafa Serkan Abdüsselam[1]
Ebru Turan-Güntepe[1]

[1]Giresun University

Abstract

This study aims to determine the perceptions of undergraduates, who are receiving coding in a faculty of education, on modal representations employed in the teaching process and identify their transition skills between representations. The research used the quantitative research method, non-experimental design, and descriptive search models, calculating the obtained data frequencies by numerical analysis. The study was carried out with the participation of 58 undergraduates in the Computer and Instructional Technology Department of an education faculty in the 2018-2019 academic year. The representational skill-testing used in the study consists of 12 open-ended questions developed by the researchers. The reliability of the test was calculated as .96 with the Pearson product-moment correlation coefficient value. Transitions between the representation of mathematics, verbal, flowchart, and code were rankly listed in the test, which was applied in a single session. The obtained data were scored with a grading key and undergraduate achievement was assessed according to the transition between representations. The analysis has revealed that representation transition skills may differ from each other and that coding teaching, which takes into account these transition skills, should be carried out with flow chart, verbal, mathematical and ultimately code representations, respectively.

Keywords: coding, modal representation, perception, representational skill testing.
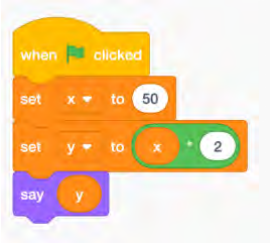
1. Introduction

The rising demand for coding teaching, combined with new skills required by our age, has made the lessons related to coding teaching both in schools (Sterritt, Hanna & Campbell, 2015; Williams et al., 2020) and on online learning platforms (Lau & Yuen, 2011; Çakıroğlu et al., 2016; Zinovieva et al., 2021) the focus of attention. Besides, coding has been introduced to curriculums from the preliminary education levels to accelerate countries' transition from consumer to producer through programs and software (Saeli et al., 2011; Demirer & Sak, 2016). Considering that computer programs do not exist without algorithms, algorithms in the learning process have become essential with the spread of computer applications (Levitin, 2012). This teaching process requires some essential skills such as program algorithm design and coding with a specific assembly code (Skiena, 2020). Moreover, the analysis skills of an algorithm will also help identify ways to make the most of the programming environment (Sedgewick & Flajolet, 2013; Gülbahar & Karal, 2018). During this process, learners are also expected to acquire basic skills such as "designing, writing, testing, debugging, and maintaining a source code of computer" program (Shadiev et al., 2014). Furthermore, it is key to coding learners to know the assembly code rules, and above all, to be able to predict solutions when encountered any problem (Renumol et al., 2009).

Any problem-solving process in coding should consists of: understanding the problem, designing the solution frame, presenting the solution through algorithms prepared with cascaded schemes, implementing the planned solution, then testing the solution, fixing it if necessary (Eker, 2005; Levitin, 2012). Learners may use various

modal representations during this process. Representations can be defined as graphics, visuals, diagrams, maps, tables, written and verbal languages (Pineda & Garza, 2002). Representations are also employed in coding teaching since they are effective for learners to experience an active learning process and to achieve substantial learning level (Seeger, 1996; Günel et al., 2009). Representations used in various fields may differ according to disciplines (Meij & Jong, 2006; Owens & Clements, 1998). Therefore, diverse representations like algorithms and flow charts are preferred in coding teaching and coding process, respectively. The use of flowcharts is vital to track the problem flow, and especially beginners in coding are recommended to prepare the program algorithm, subsequently writing the software (Eker, 2005; Kim & Lee, 2021). The last stage is to write software using assembly codes, according to the created algorithm (Ensmenger, 2016). Table 1 shows several examples of representations used by undergraduates in the algorithm and coding process during coding teaching.

Table 1. Algorithm and coding steps in the coding process

Algorithm			Coding	
Simple English	Flowchart	Pseudocode	Block-based	Text-based

Begin
Assign 50 to X
Multiply X by 2 and
Assign to Y
Write Y
End
	X,Y
	X=50
	Y=X*2
	Y
		Begin
		X=50
		Y=X*2
		Write Y
		End
			when clicked
			set x to 50
			set y to x * 2
			say y
				using System;
				public class Program
				{
				    public static void Main()
				    {
				        int X, Y ;
				        X = 50;
				        Y = X * 2;
				        Console.WriteLine(Y);
				    }
				}

Representations have different content (Table 1) and each can be used to present information distinctively (Ainsworth, 2006). Using various representations during the learning process facilitates undergraduates' learning, summarizing content, increasing the retention, and recalling of any edited display (Bodur, 2010). Undergraduates may display and manipulate contents through texts, flowcharts, shapes, mathematical expressions, or codes during the coding teaching. The role and contribution of flowcharts among these representations in coding teaching have been previously reviewed (Shneiderman et al., 1977). These charts have been regarded as a tool in coding teaching (Nickerson, 1995) and have played a key role in solving the problems encountered during the code learning-teaching (Robins et al., 2003). Coding teaching is still perceived as a difficult process for students (Porter & Kalder, 2004; Baist & Pamungkas, 2017), while continuing to pose a major challenge for instructors (Barr & Guzdial, 2015; Sáez-López et al., 2016). The effectiveness of the used representations should be examined to overcome the difficulties encountered in coding teaching processes such as learners' adaptation to programming, creating a mental model of programming (Salleh et al.2018), understanding syntax and code structure (Salleh et al.2018), debugging (Sheard et al., 2009), grasping the program structure (Lahtinen et al., 2005), and building loops (Ginat, 2004). The coding teaching process, which is structured according to the effects of these representations, is believed to contribute to the learners' overcoming the difficulties they face and becoming successful in the process.

Studies in the literature suggest the use of different teaching approaches in coding teaching. One of these approaches is the coding approach with seven steps: understand the problem, devise a plan, compare the strategies, devise an algorithm, code the algorithm, identify and correct an error in a different code and prepare and code new algorithms (Erümit et al., 2019). Additionally, it is known that the use of self-regulation strategies (Çakıroğlu et al., 2018) in the problem-solving process (Han & Kim, 2016; Yağcı, 2018; Abdüsselam et al., 2021) is also an essential part of the coding process. Although the literature studies on representations that we frequently utilize in the coding teaching process are limited, we have observed that existing studies focus on only a single representation, and that they are mostly related to flowcharts, and code representation (Gajewski, 2018; Kuljis & Baldwin, 2000). In fact, the coding teaching process requires learners to express the software they designed with flow charts, verbal expressions, and pseudo-codes in the program they use, as well as mathematical representations (Sedgewick & Flajolet, 2013). Again, the use of different modal representations and the creation of different schemas of the same information indicate the existence of transitions between modal representations (Stern

et al., 2003; Marton & Tsui, 2004). These transitions also show that a single modal representation cannot be sufficient to represent some information or situations (Lemke, 1998). Also, the positive changes in the learning of students who prefer a method suitable for different learning styles (Safari & Hejazi, 2017) feature a detailed examination of the representation types used in coding teaching. The literature review has not yielded any positive result regarding which representations should be used in each step of the coding teaching process (Skiena, 2020). In this context, it is believed that the study will contribute a lot to the education of prospective teachers, who are expected to become programmers or coding teachers in the future, revealing the role and effectiveness of using different representations in coding teaching. Against this background, we aimed to determine undergraduates' (i.e., teacher candidates') perception of the representations in the coding teaching process and their transition skills between representations. To this end, the study sought to answer the following questions:

• How are the teacher candidates' transition skills regarding mathematical expressions' transition into code, flowchart, and verbal expression?

• How are the teacher candidates' transition skills regarding verbal expressions' transition into code, mathematical expressions, and flowchart?

• How are the teacher candidates' transition skills regarding flowchart's transition into a verbal, mathematical expression, and code?

• How are the teacher candidates' transition skills regarding codes' transition into a flowchart, verbal, and mathematical expression?

• How is the teacher candidates' ability to achieve transitions between representations?

2. Method

2.1 Research Desing

The research has been done by using quantitative research method, design that is non-experimental, and descriptive research and the data obtained from it have been analyzed by the researchers (McMillan & Schumacher, 2006). Frequency analysis was used in the numerical analysis of the data and the results were presented quantitatively.

2.2 Participant

This research was conducted with 58 sophomores at the Computer Education and Instructional Technology (CEIT) Department (in the Faculty of Education) of a state university in the Eastern Black Sea Region in the spring semester of the 2018-2019 academic year. These undergraduates, who have successfully completed the Programming Languages-I course and enrolled in the Programming Languages-II, have a basic knowledge of C# programming language. Participant undergraduates who take the Programming Languages-I course in the first semester of the same academic year have sufficient knowledge of programming concepts like variables, loops, and functions. During the Programming Languages-II course, they are expected to develop software through the C# programming fundamentals they have acquired.

2.3 Data Collection Tools

A representational skill testing (RST) with 12 open-ended questions was created during the development of the data collection tool in the research. These questions were selected from the questions contained in the documents that can be used in the teaching process of the C# programming language course of the CEIT department and among those that undergraduates described as easy, medium, and difficult during five years. Afterward, feedback on the questions were collected from three programming teaching experts and a representation expert. Based on the feedback from the experts, required modifications were made for the sake of the clarity, comprehensibility, and quality of the questions. To ensure the content validity of this test, the researchers consulted two experts on instructional technologies from the university. At the same time, they collected the opinion of an expert on measurement and evaluation to determine the test conformity. The pilot test was also carried out with volunteer junior undergraduates (N=6) who completed the programming courses of the CEIT department. The test with modal representations was refined and finalized according to the feedbacks from the experts' and undergraduates' observations during the pilot study. The data obtained from RST in the main study were scored independently by two researchers. The Pearson product-moment correlation coefficient value was calculated for the sake of the consistency between the two scorings to determine the

reliability of the study. Pearson product-moment correlation coefficient value was calculated as .96 (r = 0.963, p <0.01), and it was assessed that this value represents a significant, positive, and high-level relationship between the two scorings. We can thus assume that the obtained scoring is sufficiently reliable.

The developed test is planned to be applied in 100 minutes. The questions in the test were selected and designed in a way that requires analyzing a program based on a problem with different modal representations. The main purpose is to examine undergraduates' perceptions and use of various modal descriptions to solve a problem in the text-based coding process. Having been graded as easy, medium, and difficult, the RST questions cover mathematical, verbal, flowchart, and code representations. The questions themes in modal representations, from easy to the complex are as follows: in math questions; basic calculus, arithmetic-geometric sequence, basic permutations, combinations & probability, in verbal questions; knowledge level, comprehension level, and analysis levels, in flowchart and code questions; simple flows, conditional flows, and loop structures. The implementation process of the research is schematized in Figure 1 accordingly.
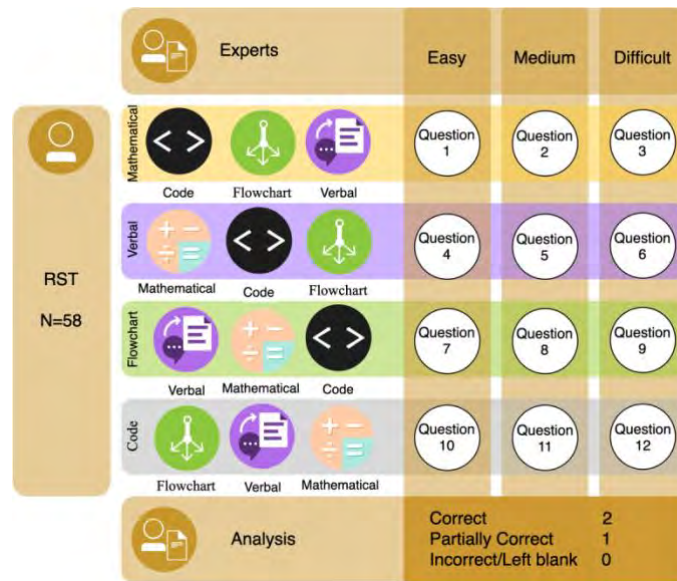

Figure 1. The research's implementation process and the grading key

RST consists of 12 questions as shown in Figure 1. RST consists of four-dimensional question representations: mathematical, verbal, flowchart, and code. In each dimension, a problem was defined with 3 questions classified from easy to difficult. The undergraduates were asked to define the algorithm and coding they will employ in the solution by using other representations. An example of RST questions is shown in Table 2.

Table 2. Examples of RST modal representation types and questions

Type	Sample Question		
	Create the multiplication table of a number to be entered externally. Finalize the relevant question with C # assembly codes, flowchart, and mathematical representations.		
	Mathematical	**Code**	**Flowchart**
	$$\sum_{i=1}^{10} sayi \cdot i$$	```csharp private void button1_Click(object sender, EventArgs e) { int sayi, islem; sayi = Convert.ToInt32(textBox1.Text); listBox1.Items.Add(sayi+" Sayısının çarpın tablosu"); for (int i = 1; i <= 10; i++) { islem = sayi * i; listBox1.Items.Add(sayi+"*"+i+"="+islem); } } ```	

2.4 Data Collection and Analysis

The open-ended questions in the RST were analyzed according to the grading key. Qualitative data were transformed into quantitative data and were analyzed through descriptive statistical processes. Researchers created the grading key, revising and amending, if necessary, both the questions and it, in line with the feedbacks from three programming teaching experts. Figure 1 shows the grading key. For the answers that were scored completely correctly, due attention was paid to whether the required answer was correctly established with the required field information, calculations, and connections, and then these answers are scored as "2". Partially correct answers that are scored as '1' are those in which the required field information, calculations, syntax error and connections are partially correct. The questions answered incorrectly or left blank were scored as "0" in the grading key.

3. Results

3.1 RQ1, How are the teacher candidates' transition skills regarding mathematical expressions' transition into code, flowchart, and verbal expression?

The levels of teacher candidates' transition of mathematical expressions into code, flow chart, and verbal expression, as well as the number of the candidates at respective levels, are given in Chart 1.
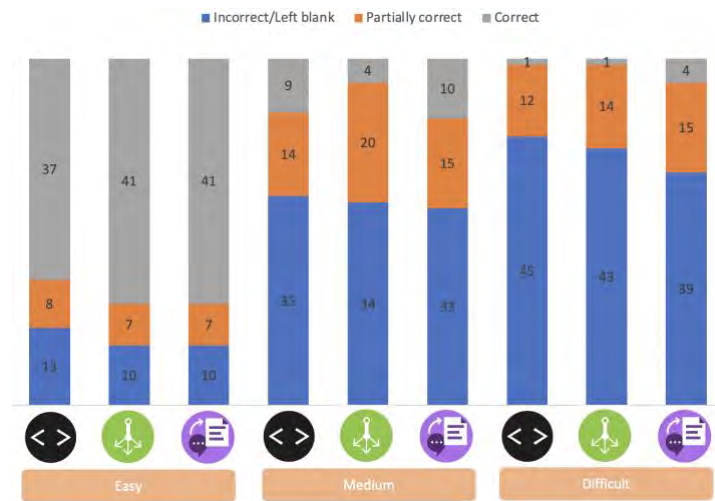

Chart 1. Transition frequencies of mathematical expressions into code, flowchart, and verbal expression

The candidate teachers were asked to present the code, flow chart, and verbal expressions of the relevant question using the mathematical expressions given in the questions. The examination of the answers given to Question 1, Question 2, and Question 3 prepared in easy, medium, and difficult levels, respectively, revealed that 37 candidate teachers in Question 1, 9 in Question 2 and 1 in Question 3 converted the mathematical expression into a code completely correctly. 41 teacher candidates in Question 1, 4 in Question 2, and 1 in Question 3 converted the mathematical expression into a flow chart completely correctly. 41 teacher candidates in Question 1, 10 in Question 2, and 4 in Question 3 were able to convert the mathematical expression into verbal expression completely correctly. Chart 1 shows the scoring distributions for each representation transition. Accordingly, for easy, medium, and difficult level questions, students' inability, or ability to realize the transitions completely and partially at each level shows a parallel distribution. Among the levels, the medium level seems closer to the difficult one. For Question 1, the candidate teachers converted the mathematical expression into flow chart, and verbal expression better than they converted it into the code. For Question 2 and 3, most of the candidate teachers were unable to convert or incorrectly converted the mathematical expression into code, flowchart, and verbal expression.

3.2 RQ2, How are the teacher candidates' transition skills regarding verbal expressions' transition into code, mathematical expressions, and flowchart?

The levels regarding the verbal expressions' transitions into code, mathematical expression and flow chart, together with the number of the candidate teachers in relevant levels, are presented in Chart 2.
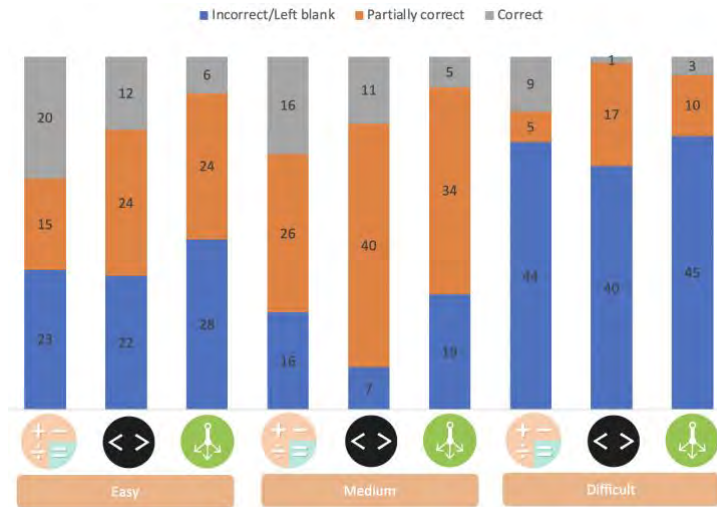
Chart 2. Transition frequencies of verbal expressions into code, mathematical expression, and flowchart

The candidate teachers were asked to present the code, flow chart and mathematical expressions of the relevant question using the verbal expressions given in the questions. The examination of the answers given to Questions 4, 5, and 6 prepared in easy, medium, and difficult levels, respectively, revealed that 12 candidate teachers in Question 4, 11 in Question 5 and 1 in Question 6 converted the verbal expression into a code completely correctly. Twenty teacher candidates in Question 4, 16 in Question 5, and 9 in Question 6 converted the verbal expression into a mathematical expression completely correctly. 6 teacher candidates in Question 4, 5 in Question 5, and 3 in Question 6 were able to convert the verbal expression into flow chart completely correctly. Chart 2 shows the scoring distributions for each representation transition. Accordingly, for easy, medium, and difficult level questions, students' inability, or ability to realize the transitions completely and partially at each level shows a parallel distribution. Among the levels, the easy and medium levels are similar in that undergraduates do the transitions partially. For Question 5, most of the teacher candidates answered the problem's code, mathematical expression, and flow chart partially correctly by using verbal expressions. In Question 4, most of the candidate teachers partially converted the verbal expression into code but could not convert it or converted it incorrectly into mathematical expression and flow chart. Most of the teachers could not convert the verbal expression into code, mathematical expression, and flow chart in Question 6.

3.3 RQ3, How are the teacher candidates' transition skills regarding flowchart's transition into a verbal, mathematical expression, and code?

The levels regarding the flow charts' transitions into a verbal and mathematical expression, and code, together with the number of the candidate teachers in relevant levels, are presented in Chart 3.
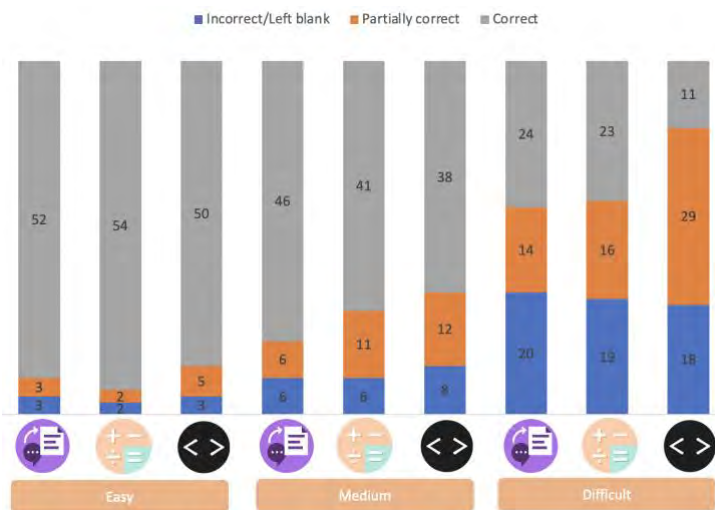

Chart 3. Flowcharts transitions to verbal, mathematical and coding expressions

The candidate teachers were asked to present the code, mathematical and verbal expressions of the relevant question using the flow charts given in the questions. The examination of the answers given to Question 7, 8, and 9 prepared in easy, medium, and difficult levels, respectively, revealed that 50 candidate teachers in Question 7, 38 in Question 8, and 11 in Question 9 converted the flow charts into a code completely correctly. Fifty-four teacher candidates in Question 7, 41 in Question 8, and 23 in Question 9 converted the flow charts into a mathematical expression completely correctly. Fifty-two teacher candidates in Question 7, 46 in Question 8, and 24 in Question 9 were able to convert the flow charts completely correctly into verbal expressions. Chart 3 shows the scoring distributions for each representation transition. Accordingly, for easy, medium, and difficult level questions, students' inability, or ability to realize the transitions completely and partially at each level shows a parallel distribution, while showing a sequential one among the levels. For Question 8, most of the candidate teachers converted the flow charts into verbal expression better than they converted it into the mathematical expression and code. In Question 7, most of the teachers converted the relevant question into code, mathematical and verbal expression completely correctly by using flow charts. For Question 9, most of the candidate teachers transitioned the flow chart into mathematical and verbal expressions, while 29 of them partially converted it to code.

3.4 RQ4, How are the teacher candidates' transition skills regarding codes' transition into a flowchart, verbal, and mathematical expression?

The levels regarding codes' transitions into a flow chart, mathematical and verbal expression, together with the number of the candidate teachers in relevant levels, are presented in Chart 4.
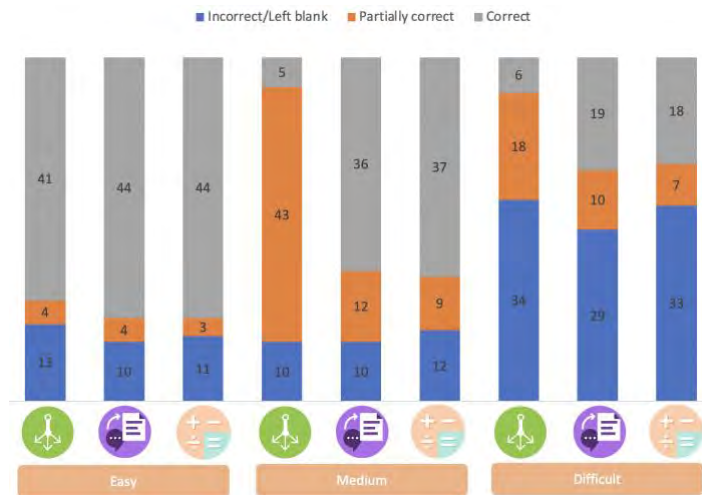

Chart 4. Codes transitions to flow chart, verbal, and mathematical expressions

The candidate teachers were asked to present the flow chart, mathematical and verbal expressions of the relevant question using the questions' code. The examination of the answers given to Questions 10, 11, and 12 prepared in easy, medium, and difficult levels, respectively, revealed that 41 candidate teachers in Question 10, 5 in Question 11, and 6 in Question 12 transferred the code to the flow chart completely correctly. Fourty-four teacher candidates in Question 10, 37 in Question 11, and 18 in Question 12 converted the code into a mathematical expression completely correctly. Fourty-four teacher candidates in Question 10, 36 in Question 11, and 19 in Question 12 were able to convert the code completely correctly into verbal expressions. Chart 1 shows the scoring distributions for each representation transition. Accordingly, for easy, medium, and difficult level questions, students' inability, or ability to realize the transitions completely and partially at each level shows a parallel distribution (excluding flowchart's transition), while the flowchart transition between levels differentiates the structure. Most of the candidate teachers converted the code into a flow chart, mathematical and verbal expression in Question 10. For Question 11, most of the candidate teachers transitioned the code into mathematical and verbal expressions, while 43 of them partially converted it to a flow chart. In Question 12, most of the candidates were unable to convert or incorrectly converted the code into flow chart, mathematical and verbal expression.

3.5 RQ5, How is the teacher candidates' ability to achieve transitions between representations?

The undergraduates' representations performed under the study were examined, and their ability to fully complete the requested transitions in 12 questions was analyzed. The obtained scores are listed in total in Table 3.

Table 3. Representations and transitions used in the research.

Questions Representations	Undergraduate's Representations				Total$_{Analysis}$
(math)		81	46	55	182
(code)	99		52	99	250
(flowchart)	118	99		122	339
(verbal)	45	24	14		83
Total$_{Rendering}$	262	204	112	276	

Table 3 shows that undergraduates have obtained different scores after converting the question representations given to them into the other three ones. The examination of the question representations specified in the table horizontally shows that the representation transition in which the candidates are the most successful in the analysis is one of those made from the flow chart (339), according to the total scores calculated. Therefore, applying the questions to the students through flow charts in coding teaching can create an added value. Lastly, examining the representations, shown vertically in the table, created by the undergraduates of the faculty of education according to the questions showed that the undergraduates were the most successful in verbal transformation (276). In teaching coding, undergraduates should be asked to analyze verbally first because this is the mode of representation they are most successful in.

4. Discussion

The research aims to reveal the relevance and effectiveness of different representations used in coding teaching and has tried to answer to how the transition skills between each representation employed in the said teaching process are transitioned into other representations. This study showed that candidate teachers' skills of expressing representations used in coding teaching may differ. The examination of mathematical expressions' transition to code, flowchart, and verbal expressions affirms that in terms of all transformations, while the candidate teachers converted easy questions, they either failed to transition or incorrectly transitioned the medium and difficult ones. The expectation that the implementation in the teaching process should progress from easy to difficult (Demirel, 2007) supports the study results. Therefore, we can assume that, as is the case in all teaching processes, simple questions that every candidate teacher can easily apprehend should be included in the curriculum, and then the desired content should be taught using more difficult questions or topics. In addition, Rizvi, Humphries, Major, Jones, and Lauzun (2011) discovered that although the number of undergraduates in computer science departments has increased in recent years, they have a weak knowledge of mathematics. Similarly, undergraduates of CEIT departments in Turkey have relatively low mathematics scores, and therefore they take Mathematics I and II courses before the programming course. In this sense, although the mathematical expressions in RST are included in the curriculum of Mathematics I and II, the fact that these expressions could not be converted or incorrectly converted to other transitions may be associated with the insufficient mathematics background of undergraduate students and the fact that the courses taken are not effective enough to improve their background.

The review of the candidate teachers' transition skills regarding converting verbal expressions into a code, mathematical expression, and flowchart has proved that significant hardship was encountered in all transitions and transitions at all relevant levels. However, the fact that candidate teachers have transitioned the problems presented by a verbal expression into the other ones can be explained by the fact that the transitions with verbal expressions in coding teaching were not solved as expected. Besides, according to Table 3, the undergraduates' presentation of the problems represented with verbal expression, their daily communication representation, more successfully than transitions in other representations reveals that the initial representation they will use in programming teaching should be verbal expressions. In such cases, Lemke (1993, 1998) specifies that students should be assisted to

establish appropriate verbal connections with mathematical expressions, diagrams, graphics, and all the other representations. Thus, candidate teachers would be able to think beyond the box and have critical thinking with an effective questioning approach (Crafton et al., 2009; Kazimoglu et al., 2012).

The review of the candidate teachers' transition skills regarding converting flow charts into codes, verbal and mathematical expressions has revealed that there is a significant success in easy, medium, and difficult transition levels in terms of all transformations. Visualizing the coding teaching process that is perceived as complex by learners, thereby making it more apprehensible, can explain this phenomenon. In that vein, Hagevik, Beilfuss, and Dickerson (2006) stated that modal representations, which form a visual state of knowledge, simplify complex meanings, and organize individuals' cognitive processes by supporting them. It is still an expected phenomenon that teacher candidates are more likely to be unable to do or make mistakes with difficult questions when compared to the other two levels during the process of flow charts' transformations. Still, we assert that the transition to codes at this level can be performed rather less compared to the other transitions. In fact, it is remarkable that few candidate teachers have completely correctly answered the transformations in flow charts compared to other transformations, during the teaching-learning process, where the transition from flow charts to codes is often made in the coding teaching. Nonetheless, several studies in the literature suggest that coding should be realized following the creation of the flowchart (Ergin & İpek, 2017; Gajewski, 2018; Türker & Pala, 2020). Accordingly, we assert that a difficult question on flow charts is more distinctive compared to questions regarding the other transitions. Also, the examination of the candidate teachers' transition skills regarding the converting codes into flow charts, verbal and mathematical expressions has shown that learners experienced difficulties in these representation transitions, especially in flow chart transitions compared to the others (Chart 4). Among these transitions, we have found that candidate teachers had problems in transitions from flow chart to code and vice versa.

Remarkably, the transition from flow chart to verbal and mathematical expression and codes is at the forefront according to candidate teachers' ability to perform all transitions. Additionally, in these transitions, what candidates can do best in each representation again matches the same transition. In general, coding teaching requires candidate teachers to set up an algorithm and then convert it into coding. The lecturer may exploit verbal and mathematical expressions or flowcharts during the process of creating an algorithm. The research results showed that by presenting the modal representations in coding teaching, the learners understood the program most successfully and were able to re-transition it into the other representations. Accordingly, coding teaching should be started with flow charts designed in shapes related to the program. In this sense, the verbal representation is the most successful one for candidate teachers in the case of re-representing the curriculum. Therefore, presenting flowcharts to learners and having discussions with them about the program can be regarded as the second step of coding teaching. The next step is to describe the program to be taught through mathematical expressions according to the obtained results. The last step of this process is code representations following the mathematical presentation. As Gajewski (2018) stated, flowcharts are rather effective tools for coding teaching and play an essential bridging role in the algorithm creation and then in the coding execution. Also, Kuljis and Baldwin (2000) affirm that the formal representations used under the scope of a flow chart for learners who are new to coding are highly suitable for the coding teaching. According to Ramadhan (2000), the reason behind this is that supporting (learning) topics related to decision structures and loops in coding teaching with visual tools like shapes has a positive effect on the success. However, as Hu (2004) stated, theories and practical applications should be handled simultaneously in coding teaching. Similarly, this research has also shown that flowcharts are seized more easily by coding learners then the other representations and that transition to the other representations is more successful.

This study has been carried out for the use of text-based coding tools at the undergraduate level. In this context, we suggest examining the relevance and effectiveness of these representations during the use of block-based tools. We also recommend that the coding teaching process be structured according to the order of use of the mathematical, verbal, flowchart, and code representations specified in the research. Finally, future studies may examine relationships between various variables such as the effectiveness of the newly developed representation usage order, and learners' attitudes and their programming self-efficacy.

References

Abdüsselam, M. S., Turan-Güntepe, E., & Durukan, Ü. G. (2021). Investigation of the Teaching Process of Programming in regards to Problem Solving Process and Perception of Self-Efficacy. *Journal of Bayburt Education Faculty*, *16*(31), 149-173.

Ainsworth, S. (2006). DeFT: A conceptual framework for considering learning with multiple representations. *Learning and instruction*, *16*(3), 183-198. https://doi.org/10.1016/j.learninstruc.2006.03.001

Aslanyürek M., Korkmaz A., Büyükgöze S.B. & Gezgin D.M (2018) *Algorithm and Programming All Resolved Question Bank*, Efeakademi Publisher.

Baist, A., & Pamungkas, A. S. (2017). Analysis of Student Difficulties in Computer Programming. *VOLT: Jurnal Ilmiah Pendidikan Teknik Elektro*, *2*(2), 81-92. https://doi.org/10.30870/volt.v2i2.2211

Barr, V., & Guzdial, M. (2015). Advice on teaching CS, and the learnability of programming languages. *Communications of the ACM, 58*(3), 8–9.

Bodur, F. (2010). *Additives to learning visual elements in distance teaching textbooks: Evaluation of Anadolu University distance teaching student views*. Eskişehir: Anadolu University.

Çakıroğlu, Ü., Kokoç, M., Kol, E., & Turan, E. (2016). Exploring teaching programming online through web conferencing system: The lens of activity theory. *Journal of Educational Technology & Society*, *19*(4), 126-139.

Çakıroğlu, Ü, Er, B. Uğur, N., & Aydoğdu, E. (2018). Exploring the use of self-regulation strategies in programming with regard to learning styles. *International Journal of Computer Science Education in Schools, 2*(2), 14-28. https://doi.org/10.21585/ijcses.v2i2.29

Clark, J. M., & Paivio, A. (1991). Dual coding theory and education. *Educational psychology review*, *3*(3), 149-210. https://doi.org/10.1007/BF01320076

Crafton, L., Brennan, M., and Silvers, P. (2009). Creating a critical multiliteracies curriculum: Repositioning art in the early childhood classroom. Making Meaning.

Crafton, L. K., Silvers, P., & Brennan, M. (2009). *Creating a critical multiliteracies curriculum: Repositioning art in the early childhood classroom. In Making meaning*. Springer.

Demirel, Ö. (2007). *Teaching principles and methods: The art of teaching*. Ankara: Pegem A Yayıncılık.

Demirer, V., & Sak, N. (2015). Information and communication technology (ICT) education in Turkey and changing roles of the ICT teachers. *The Journal of International Education, 2* (5), 434-448.

Du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, *2*(1), 57-73. https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9

Eker M. (2005), *Understanding the algorithm*, Niravan Yayınları, Ankara

Ensmenger, N. (2016). The multiple meanings of a flowchart. *Information & Culture*, *51*(3), 321-351. https://doi.org/10.7560/IC51302

Ergin, H., & İpek,J (2017). Collaborative Creative Problem-Solving Model in Programming Language Teaching: A Case Study. *Ege Journal of educational Technologies*, *1*(2), 135-148.

Erümit, K. A., Karal, H., Şahin, G., Aksoy, D. A., Aksoy, A., & Benzer, A. I. (2019). A model suggested for programming teaching: Programming in seven steps. *Education and Science, 44*(197), 155-183. http://dx.doi.org/10.15390/EB.2018.7678

Gajewski, R. R. (2018). Algorithms, Programming, Flowcharts and Flowgorithm. *E-Learning and Smart Learning Environment for the Preparation of New Generation Specialists*, 393-408.

Ginat, D. (2004). On novice loop boundaries and range conceptions. *Computer Science Education, 14*(3), 165-181.

Gülbahar, Y., & Karal, H. (2018). *Teaching programming from the theory to application [Kuramdan uygulamaya programlama öğretimi]*. Ankara: Pegem Akademi Yayıncılık.

Günel, M., Atila, M. E., & Buyukkasap, E. (2009). The impact of using multi modal representations within writing to learn activities on learning electricity unit at 6th grade. *Elementary Education Online*, *8*(1), 183-199. http://ilkogretim-online.org.tr/index.php/io/article/viewFile/1705/1541

Hagevik R, Beilfuss M, Dickerson D (2006). Multiple representation sin science education. Paper presented at the annual meeting of the National Association for Research in Science Teaching (NARST), April 3–6, San

Francisco

Han, S. J., & Kim, S. S. (2016). The effects of app programing education using m-Bizmaker on creative problem solving ability. *The Journal of Korean Association of Computer Education*, *19*(6), 25-32.

Hu, M. (2004). Teaching novices programming with core language and dynamic visualisation. *Proceedings of the 17th NACCQ*, 94-103. Retrieved from https://www.citrenz.ac.nz/conferences/2004/hu.pdf

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, *9*, 522-531. https://doi.org/10.1016/j.procs.2012.04.056

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, *37*(2), 83-137. https://doi.org/10.1145/1089733.1089734

Kim, Y., & Lee, M. (2021). Development of an unfolding model of procedures for programming learning of novice programmers. *Computer Applications in Engineering Education*. 1-20. https://doi.org/10.1002/cae.22437

Kuljis, J., & Baldwin, L. P. (2000). Visualisation Techniques for Learning and Teaching Programming. *Journal of computing and information technology*, *8*(4), 285-291. https://doi.org/10.2498/cit.2000.04.03

Lahtinen, E., Ala-Mutka, K. & Jarvinen, H. (2005) A Study of Difficulties of Novice Programmers. *In Acm Sigcse Bulletin, ACM, 37*(3), 14-18.

Lau, W. W., & Yuen, A. H. (2011). Modelling programming performance: Beyond the influence of learner characteristics. *Computers & Education*, *57*(1), 1202-1213. https://doi.org/10.1016/j.compedu.2011.01.002

Law, K. M., Lee, V. C., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, *55*(1), 218-228. https://doi.org/10.1016/j.compedu.2010.01.007

Lemke, J. L. (1993, December 17-18). *Multiplying meaning: Literacy in a multimedia world* [Conference presentation abstract]. Annual Meeting of the National Reading Conference, Charleston, SC, United States.

Lemke, J. (1998). *Multiplying meaning: Visual and verbal semiotics in scientific text*. London: Routledge.

Levitin, A. (2012). *Introduction to design and analysis of algorithm*. Pearson, Boston.

Marton, F., & Tsui, A. (2004). *Classroom discourse and the space of learning*. Mahwah, NJ: Erlbaum

McMillan, J. H., and Schumacher, S. (2006). *Research in Education: Evidence-based Inquiry*. Cape Town: Pearson.

Meij, J., & de Jong, T. (2006). Supporting students' learning with multiple representations in a dynamic simulation-based learning environment. *Learning and instruction*, *16*(3), 199-212. https://doi.org/10.1016/j.learninstruc.2006.03.007

Nickerson, J. V. (1995). *Visual programming*. New York, NY: New York University.

Owens, K. D., & Clements, M. K. (1998). Representations in spatial problem solving in the classroom. *The Journal of Mathematical Behavior*, *17*(2), 197-218. https://doi.org/10.1016/S0364-0213(99)80059-7

Pineda, L., & Garza, G. (2000). A model for multimodal reference resolution. *Computational Linguistics*, *26*(2), 139-193. https://doi.org/10.1162/089120100561665

Porter, R., & Calder, P. (2004, January). Patterns in learning to program: an experiment?. In *Proceedings of the Sixth Australasian Conference on Computing Education-Volume 30* (pp.241-246). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.2526&rep=rep1&type=pdf

Qian, Y., & Lehman, J. (2020). An Investigation of High School Students' Errors in Introductory Programming: A Data-Driven Approach. *Journal of Educational Computing Research*, *58*(5), 919-945. https://doi.org/10.1177/0735633119887508

Ramadhan, H. A. (2000). Programming by discovery. *Journal of Computer Assisted Learning*, *16*(1), 83-93. https://doi.org/10.1046/j.1365-2729.2000.00118.x

Renumol, V., Jayaprakash, S., & Janakiram, D. (2009). Classification of cognitive difficulties of students to learn computer programming. *Indian Institute of Technology, India*, *12*. http://dos.iitm.ac.in/publications/LabPapers/techRep2009-01.pdf

Rizvi, M., Humphries, T., Major, D., Jones, M., & Lauzun, H. (2011). A CS0 course using scratch. *Journal of Computing Sciences in Colleges, 26*(3), 19-27.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education*, *13*(2), 137-172. https://doi.org/10.1076/csed.13.2.137.14200

Saeli, M., Perrenet, J., Jochems, W. M., & Zwaneveld, B. (2011). Teaching programming in Secondary school: A pedagogical content knowledge perspective. *Informatics in education*, *10*(1), 73-88. https://doi.org/10.15388/infedu.2011.06

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

Safari, E., & Hejazi, M. (2017). Learning styles and self-regulation: an associational study on high school students in iran. *Mediterranean Journal of Social Sciences, 8*(1), 463. https://doi.org/10.5901/mjss.2017.v8n1p463

Salleh, S. M., Shukur, Z., & Judi, H. M. (2018). Scaffolding model for efficient programming learning based on cognitive load theory. *Int. J. Pure Appl. Math*, *118*(7), 77-83. https://acadpubl.eu/jsi/2018-118-7-9/articles/7/10.pdf

Sedgewick, R., & Flajolet, P. (2013). *An introduction to the analysis of algorithms*. Pearson Education India.

Seeger, F. (1996). *Representations in the mathematics classroom: Reflections and constructions*. University of Helsinki.

Shadiev, R., Hwang, W. Y., Yeh, S. C., Yang, S. J., Wang, J. L., Han, L., & Hsu, G. L. (2014). Effects of unidirectional vs. reciprocal teaching strategies on web-based computer programming learning. *Journal of educational computing research*, *50*(1), 67-95. https://doi.org/10.2190/EC.50.1.d

Sheard, J., Simon, S., Hamilton, M., & Lönnberg, J. (2009, August). Analysis of research into the teaching and learning of programming. In *Proceedings of the fifth international workshop on Computing education research workshop* (pp. 93-104). https://doi.org/10.1145/1584322.1584334

Shneiderman, B., Mayer, R., McKay, D., & Heller, P. (1977). Experimental investigations of the utility of detailed flowcharts in programming. *Communications of the ACM*, *20*(6), 373-381. https://doi.org/10.1145/359605.359610

Skiena, S. S. (2020). *The algorithm design manual*. Springer International Publishing.

Stern, E., Aprea, C. & Ebner, G. E. (2003). Improving Cross-Content Transfer in Text Processing by Means of Active Graphical Representation. *Learning and Instruction, 13* (2),191-203. https://doi.org/10.1016/S0959-4752(02)00020-8

Sterritt, R., Hanna, P., & Campbell, J. (2015, March). Reintroducing programming to the school environment. In *INTED 2015-9th International Technology, Education and Development Conference* (pp. 7630-7631). International Academy of Technology, Education and Development. Madrid, Spain. Retrieved from http://library.iated.org/view/STERRITT2015REI

Türker, P. M., & Pala, F. K. (2020). The effect of algorithm education on students' computer programming self-efficacy perceptions and computational thinking skills. *International Journal of Computer Science Education in Schools*, *3*(3), 19-32.

Williams, H. E., Williams, S., & Kendall, K. (2020, June). CS in Schools: Developing a Sustainable Coding Programme in Australian Schools. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 321-327). https://doi.org/10.1145/3341525.3387422

Yağcı, M. (2018). A Study on Computational Thinking and High School Students' Computational Thinking Skill Levels. *International Online Journal of Educational Sciences, 10*(2), 81-96. doi:10.15345/iojes.2018.02.006

Zinovieva, I. S., Artemchuk, V. O., Iatsyshyn, A. V., Popov, O. O., Kovach, V. O., Iatsyshyn, A. V., ... & Radchenko, O. V. (2021, March). The use of online coding platforms as additional distance tools in programming education. In *Journal of Physics: Conference Series* (Vol. 1840, No. 1, p. 012029). IOP Publishing. https://doi.org/10.1088/1742-6596/1840/1/012029