CONTEMPORARY EDUCATIONAL TECHNOLOGY, 2017, 8(3), 195-213

Individual Differences in Learning Computer Programming: A Social Cognitive Approach

Sacide Guzin Mazman Akar
Usak University, Turkey

Arif Altun
Hacettepe University, Turkey

Abstract

The purpose of this study is to investigate and conceptualize the ranks of importance of social cognitive variables on university students' computer programming performances. Spatial ability, working memory, self-efficacy, gender, prior knowledge and the universities students attend were taken as variables to be analyzed. The study has been conducted with 129 2nd year undergraduate students, who have taken Programming Languages-I course from three universities. Spatial ability has been measured through mental rotation and spatial visualization tests; working memory has been attained through the measurement of two sub-dimensions; visual-spatial and verbal working memory. Data were analyzed through Boosted Regression Trees and Random Forests, which are non-parametric predictive data mining techniques. The analyses yielded a user model that would predict students' computer programming performance based on various social and cognitive variables. The results yielded that the variables, which contributed to the programming performance prediction significantly, were spatial orientation skill, spatial memory, mental orientation, self-efficacy perception and verbal memory with equal importance weights. Yet, the effect of prior knowledge and gender on programming performance has not been found to be significant. The importance of ranks of variables and the proportion of predicted variance of programming performance could be used as guidelines when designing instruction and developing curriculum.

Keywords: *Improving classroom teaching; Computer programming; Social cognitive approach; Individual differences*

Introduction

Programming is a very complex and multi-staged process, where each sub-process requires a different content knowledge, demands different cognitive processes (Ambrosio, Costa, Almeida, Franco, & Macedo, 2011) and numerous skills to be addressed in teaching programming (Howard, 2002; Lehman, Bruning, & Horn, 1983). For instance, reading comprehension, critical reasoning, systematic thinking, acquiring cognitive components in problem identification, planning and producing solutions, creativity, intellectual curiosity, mathematical skills, situational reasoning, procedural thinking, temporary reasoning, analytical and quantitative reasoning, making use of different sources, being creative and flexible in producing solutions are just some of those reported skills (Ambrosio et al., 2011; Lau & Yuen, 2011).

Programming competencies are based on knowledge and skills; knowledge consists of definitions, facts, language constructs (i.e., syntax) and specific algorithms of programming knowledge whereas programming skills consists of certain required actions including the strategies in applying this knowledge (Caspersen, 2007). Individuals perform differently when they transfer their knowledge of programming knowledge into actions. Yet, whether there is any individual differences and how those differences play a role in programming remain a salient question to be explored.

In search of an answer to explore these individual differences, researchers undertake an expert-novice paradigm. The main underlying premise with expert-novice paradigm is that if we knew how experts perform and which steps they follow; one can easily observe the differences between users. Therefore, these differences help educators to make judgments about (i.e., (Ambrosio et al., 2011; Mancy & Reid, 2004), and develop instructional guidance in teaching programming for poor or unsuccessful individuals (i.e., Mason, Seton & Cooper, 2016). Research about the individuals with high programming performance in general reported that those individuals are intelligent, intellectually challenging, and able to think analytically (Byrne & Lyons, 2001). In a recent study, Lin et al. (2016) explored students' cognitive processes while debugging programs and found that expert (high-performance) students traced programs in a more logical manner.

Since the 1950s, researchers wanted to determine which variables are more effective in predicting computer programming performances (i.e., Alspaugh, 1972; Bergersen & Gustafsson, 2011; Merrienboer & Paas, 1990; Rowan, 1957), mainly using the expert-novice paradigm (Lin, Wu, Hou, Lin, Yang, & Chang, 2016). Among those variables are gender, personality, intelligence, attitude towards computers, experience, level of comfort, background in mathematics, courses taken, the playing of games (Charlton & Birkett, 1999; Wilson, 2002), academic background and psychological factors (Bergin & Reilly, 2006), cognitive, behavioral and attitudinal factors (i.e., deRaadt et al., 2005), study habits (Willman, Linden, Kaila, Rajala, Laakso, & Salakoski, 2015) and cognitive skills (i.e., Bergersen & Gustafsson, 2011). Recently, on the other hand, there is a common criticism in that individual differences have not been sufficiently quantified (Bergersen & Gustafsson, 2011), and the problem of construct validity in relation to programming performance has still not been resolved (Hannay, Arisholm, Engvik, & Sjoberg, 2010).

Learning a programming language is both social and cognitive activity. It is suggested in the literature that cognitive structures should be analyzed to understand a person's programming process (Caspersen, 2007), however the effect of cognitive skills on programming performance in studies has generally been neglected for a long time (Irons, 1982). Furthermore, students' understanding of "selves" might be a powerful predictor of their success in programming (Askar & Davenport, 2009); so, it would not be comprehensive if such social cognitive attributes are left out in analyses. With regard to instructional context, the measurement related to individuals' cognitive processes may be used as clues in developing course programs for educational purposes (Shute, 1991; Lin, et al., 2016). In addition, knowing about the factors which affect programming performance of individuals may be beneficial in supporting individuals which in advance come as disadvantaged (Byrne & Lyons, 2001). From this point of view, the purpose of this study is to explore the importance ranks of some certain cognitive and social variables when learning a programming language in an undergraduate setting.

Theoretical Background

Individual Differences: Social and Cognitive Variables

Individuals become different from each other both in terms of their general skills, preferences of knowledge processing, making inferences from knowledge, and applying them to new situations with varying learning tasks in real life situations. Different tasks, environments and outputs require different skills and abilities (Jonassen & Grabowski, 1993). In cases where the effect of cognitive individual differences are not controlled, it has been stated that the efficiency of learning processes cannot clearly be put forth (Stalcup, 2005). Cognitive differences in didactic manipulations and the individual's performance are emphasized (Ackerman, Beier, & Bowen, 2002) and taking the differences in individuals' cognitive skills into consideration is important throughout life (Alwin, 1994).

It has been well accepted that programming performance is not a sum of individual skills alone, but a hierarchy of skills where the programmer uses any of these skills at any stage (Jenkins, 2002). Furthermore, it has been underlined that cognitive skills such as perception, attention and memory capacity (Irons, 1982; Jenkins, 2002) and their interaction effects with(in) social interactions should be taken into consideration. Thus, the differences in cognitive processes are as a result of many cognitive factors among which are memory capacity, attention spans spatial skills, perceptions, language acquisition, mental models, problem solving, and reasoning. Working memory and spatial skills, nevertheless, are the most frequently reported variables when handling tasks in computer-based environments (Román-González, Pérez-González & Jiménez-Fernández, 2016; Pak, Rogers, & Fisk, 2006). In the following section, those variables will be presented briefly and reported based on the literature findings.

Spatial ability. One of the individual differences which relates to programming performance is spatial ability, which is addressed differently in the forms of verbal, mathematical and reasoning skills, as a dimension of intelligence and heterogeneous skills clusters (Jones & Burnett, 2008). Spatial skill is defined in the most general sense as perception, coding, remembrance, transformation, differentiation of symbolic and non-verbal knowledge. It can be expressed that spatial characteristics such as location, dimension, size, distance, direction, shape and movement are cognitive (Lawton, 2010; McGee, 1979).

Spatial ability is closely related to daily life experiences and is needed for many processes such as interaction with various tools, remembering-depiction of space and direction, mental visualization, making plans, etc. (Lawton, 2010). Spatial ability is also reported to be one of the major variables in the literature to determine cognitive differences (Blustein & Satel, 2003; Vicente & Williges, 1988), and academic success in many areas such as mathematics, physics, chemistry, engineering, architecture, medicine, graphics, art, computer sciences, etc. and in the choice of profession related to these areas (McGee, 1979; Wright, Thompson, Ganis, Newcombe, & Kosslyn, 2008).

Working Memory. Working memory is a system which stores and integrates information during complex activities (Baddeley, 1992). This cognitive structure has an important role in the performance of various complex cognitive tasks (Haavisto & Lehto, 2005). Working memory, with its critical roles such as temporary activation of long-term memory, coordination of multiple duties, task switching and calling for strategies, organization of capacity to be used or allocated, emerges as a structure which determines the performance of other cognitive processes (Daneman & Merikle, 1996; Mancy & Reid, 2004).

Learning a programming language requires various syntactic (syntax) skills. Code writing, code reading, knowing unique writing rules for each programming language require individuals to put their verbal memory process at work. On the other hand, although a written program consists of small code particles, it requires navigation between the code particles and processes to achieve the perception of the whole program. The correction of errors, creating a mental model for the representation of its flow and hierarchy in relation to the program's algorithm bring individual's visual-spatial memory to the fore.

Programming Self-Efficacy. The belief of self-efficacy affects the individual's choice of efficiency in being successful when performing a task, the level of the effort spent, and resistance in dealing with difficulties and most importantly, performance (Bandura, 1977). Rather than knowledge, skill and previous success of the individual in any context, the individual's belief in the results of his own skills and efforts has a very strong effect (Pajares, 1996). In this context, self-efficacy is defined as the belief of the individual in his skills and even if the individual has doubts about his performance despite sufficient knowledge and skill, if his motivation is low and his perception about being successful is low, then it is expressed that the individual may be unsuccessful (Askar & Davenport, 2009).

Prior knowledge. Previous knowledge or prior experience in a certain field mostly emerge as having a balancing effect in differences related to individuals' skills. In other words, if the individual has sufficient preliminary knowledge or experience, the previous experiences he has will facilitate the process without feeling the need to use that skill, even if some of his cognitive skills are low. In related literature, the effect of preliminary knowledge or experience on programming performance is dealt with either separately or as a single factor (Bergin & Reilly, 2005; deRaadt et al., 2005; Lau & Yuen, 2011).

Gender. One of the variables where the performances of individuals differ in computer-based environments is gender. Gender is known for having a quite variance in programming research. Since disciplines such as computer sciences are preferred mostly by males, there is a difference in the feeling of efficacy in terms of gender other than the individuals' existing skills and performances. In addition, there is a prejudiced difference had been reported in terms of males and females from an external perspective (Byrne & Lyons, 2001; Svedin & Balter , 2016).

The Current Study

When designing a course or program in teaching programming based on individuals' characteristics, decision makers can tailor the needs of individuals when presenting the content especially for novices. Therefore, this study aims at contributing to the development of a personalized model taking the social cognitive factors into account. The variables included spatial skill, working memory, self-efficacy, gender, university and prior knowledge, which have been included into the model to identify the relationship between these variables and their order of importance when developing a predictive model. More specifically, the following research questions are framed:

1. Is there a relationship between college students' programming performance, and their self-efficacy, spatial abilities and working memory capacities?
2. How much of the programming performance can be predicted by cognitive abilities, self-efficacy, prior knowledge, university and gender?

Method

This study has been modeled on the foundation of Boosted Regression Trees and Random Forests techniques that are based on mining and are used in non-parametric situations. The model has firstly been tested with Boosted Regression Trees method and with Random Forest technique in order to predict the validity of the model.

Participants

The study group consisted of 129 undergraduates attending the Department of Computer Education and Instructional Technology (CEIT) from three different universities who have taken the Programming Languages-I course. The Programming Languages-I course is the first course given in relation to programming in CEIT and is offered in the third semester. Participation to the study was on a volunteer base. All of the students are in the 18-25 age group, 69 (53.5%) are female and 60 (46.5%) male, and all have taken the first programming course in their undergraduate studies. All universities were located in the same city and were state-funded. They select their students based on a nation-wide university entrance exam and they were among the top 10% of the overall placement nationwide.

Data Collection Tools

A demographic information form, spatial orientation test, visual-spatial memory test, verbal memory test, self-efficacy perception scale related to programming and programming performance grades have been gathered through computerized measurement tools. These measurement tools are described in detail below.

Spatial Orientation Test. In the study, the computer-based “Spatial Orientation Test” has been used. This test was originally developed by Kozhevnikov and Hegarty (2001) and revised by Mazman and Altun (2013). Psychometric properties and norm data for Turkish university students was presented in detail at Mazman and Altun (2013) study.

Visuo-Spatial Memory / Number Rotation Test. “Visuo-Spatial Memory-Number Rotation Test” was developed by Blasko, Holliday-Darr, Mace, and Blasko-Drabik, (2004) within the scope of Visual Evaluation and Instruction Project (VIS) (http://viz.bd.psu.edu), to evaluate, analyze and develop spatial performances. There are two different sub-tasks in the test, namely rotating certain letters in the mind and keeping the direction locations of the rotated letters. This test has been coded in E-Prime by the researchers. Based on the participants’ performances, a visuo-spatial memory score is generated.

Verbal Working Memory Test. The measurement of the individuals’ verbal working memory has been carried out by the n-back task software called “Brain workshop” (Hoskinson, 2012). Since the software is open source code, it has been compiled and made ready to use by the translation of its interface and sound files (Cevik, 2012). The default beginning of the software is dual 2-back mode. Here, dual expresses the presentation type of the software – its modality – (sound, location, color, shape… etc.) and 2-back expresses the n value needed to go back. In the study, since only the auditory working memory was measured, only the auditory modality has been utilized.

Self-efficacy for Programming. The "Programming Self-efficacy Scale", developed by Ramalingam and Wiedenbeck (1998) and adapted to Turkish by Altun and Mazman (2012) has been used to measure perceived self-efficacy related to programming. The scale consists of nine questions with a seven Likert-type options. It consists of two factors, "complex programming tasks" and "simple programming tasks". The minimum score achievable in the scale is 7 and the maximum is 63. The internal consistency coefficient of the scale has been calculated as 0.928.

Programming Performance. For programming performance, the end of semester grades individuals received upon completion of the Programming-I course have been evaluated. The students' end of year passing grade (as percentages) has been requested from three instructors at three universities. The end of the year passing grade consists of the midterm and the end of year final tests.

Demographic Variables. Data about gender, age, university and prior knowledge were gathered through a demographic information form. Questions about prior knowledge consisted of "if they had taken any programming course before the university and whether they could write a program in any programming languages?" If answer is "yes" for either of the questions that participant is coded as having prior knowledge. If answer is "no" for both of the two questions that participants is coded as having no prior knowledge.

Data Collection Process

Data has been collected from three different universities. Once the students were informed about the place and duration of the data collection process, each session was carried out on a one-to-one basis with each student, each lasting between 25 to 30 minutes. Participants were informed about that the time was not limited but their reaction time will be logged through their number of correct answers for scoring. Instruction screens and sample solution video was presented for a computer-based test, which was followed by practice questions proceeding to the test questions. Once the computerized tests were finished, the participants were asked to fill out a self-efficacy scale related to their computer programming experience.

Scoring

The total score received from the self-efficacy scale was calculated for each participant (maximum: 63 and minimum: 0). Verbal working memory scores were computed by the software itself as a performance score, which was calculated in terms of reaction times and accuracy (See Jaeggi, 2010 for detailed information). For the Spatial Orientation, Mental Rotation, and Visuo-Spatial Memory Tests, two separate scores were calculated; the reaction time score and accuracy score. Since both the accuracy score and reaction time score present important information about the difficult level of the task and the skills individuals possess, it is stated that it would be more significant to weigh and integrate the accuracy and reaction time scores and produce a single dependent variable (Bruyer & Brysbaert, 2011; Wagenmakers et al., 2007). Thus, the general efficacy scores suggested by Luft et al. (2013) have been calculated by the following formula (1).

$$\text{Efficacy Score} = \frac{\text{Number of correct answers - Number of wrong answers}}{\text{Average Reaction Time}} \quad (1)$$

Data Analysis

Boosted Regression Tree (BRT), which is one of the tree-based methods based on data mining method, was utilized to analyze the data. According to traditional approaches, the reactions given by individuals in cognitive systems are generally the common product of the effect of a series of constructs. Although it is known that the inner dynamics of these constructs are complex, the interaction between these constructs are limited most of the time as in the most general example of the assumption, the interaction between the constructs is linear (Carello & Moreno, 2005). However, cognitive skills cannot always be linear. Therefore, there is a need for non-linear methods which could help us better define such complex relationships in particular (Luft et al., 2013). In order to test the validity of BRT findings, the same model has been analyzed with the Random Forests method.

BRT is a technique which combines the strengths of regression trees (models that relate a response to their predictors by recursive binary splits) and boosting (an adaptive method for combining many simple models to give improved predictive performance) (Elith, Leathwick, and Hastie, 2008). On the contrary of standard regression methods which produce a single predictive model, BRT fit multiple simple models and combine them for prediction, thus improving predictive performance (Buston & Elith, 2011). While different algorithms can be used to build a BRT model, Stochastic Gradient Boosting algorithm (Friedman, 2002) was performed. Random Forests, on the other hand, is a technique based on model aggregation ideas, for both classification and regression problems (Genuer, Poggi, & Tuleau-Malot, 2010). Random Forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forests (Breiman, 2001). Random Forests runs efficiently on large data bases and handle thousands of input variables without variable deletion giving estimates of what variables are important in the classification (Breiman & Cutler, 2004).

Findings

The descriptive statistical results achieved from the spatial orientation test, mental rotation test, visuo-spatial memory test, verbal memory test and self-efficacy perception scale related to the programming grades taken as programming performance are given in Table 1.

Table 1. Descriptive Statistics Related to Programming Performance, Spatial Orientation, Working Memory and Self-Efficacy Perception

	$\overline{X}$	Sd	se	Kurtosis	Skewness
Programming Performance	66.36	18.4	1.62	0.506	-0.790
Self-efficacy	42.12	11.8	1.04	-0.270	-0.562
Spatial Orientation	0.1016	0.0053	0.00047	1.900	-0.955
Mental Rotation	0.0043	0.0032	0.00027	0.509	0.633
Spatial Memory	0.0026	0.0058	0.00051	0.263	-0.142
Verbal Memory	60.64	19.1	1.68	-0.431	-0.246

The correlations between the variables has been calculated and presented in Table 2.

Table 2.Correlation Matrix between the Variables

	Programming Performance	Self-efficacy	Spatial Orientation	Mental Rotation	Spatial Memory	Verbal Memory
Progr. Perform.	1					
Self-efficacy	0.406**	1				
Spatial Orientation	0.063	0.000	1			
Mental Rotation	0.027	0.062	0.130	1		
Spatial Memory	0.117	0.091	0.168	0.061	1	
Verbal Memory	-0.037	0.059	0.089	0.148	0.225*	1

As presented in Table 2, while there is a very low (r<0.5, p<0.05) correlation between programming performance and self-efficacy perception and spatial memory and verbal memory, no correlations between the other variables have been significant.

Predicting Programming Performance Model with Tree-based Methods

In order to determine the predictive power of programming performance of a total of eight variables, the boosted regression analysis technique has been applied. The maximum number of nodes for each tree has been set to be three as the default, the number of additive terms (successive number of trees) has been left as 200 as the default and learning rate (shrinkage parameter) has been selected as 0.1000 as default. The random test data proportion has been determined as 0.1 (10%). This proportion expresses that 10% of the total observation number will be used as test data for validity and 90% will be used as education data for modeling. As a result of the analysis, the most suitable sub-sampling value has been determined as 0.45. Default values have been set for stop parameters. The analyses yielded the optimal number of trees to reach at 166. The optimization graphic related to the boosted regression tree basic effect model is given in Figure 1.

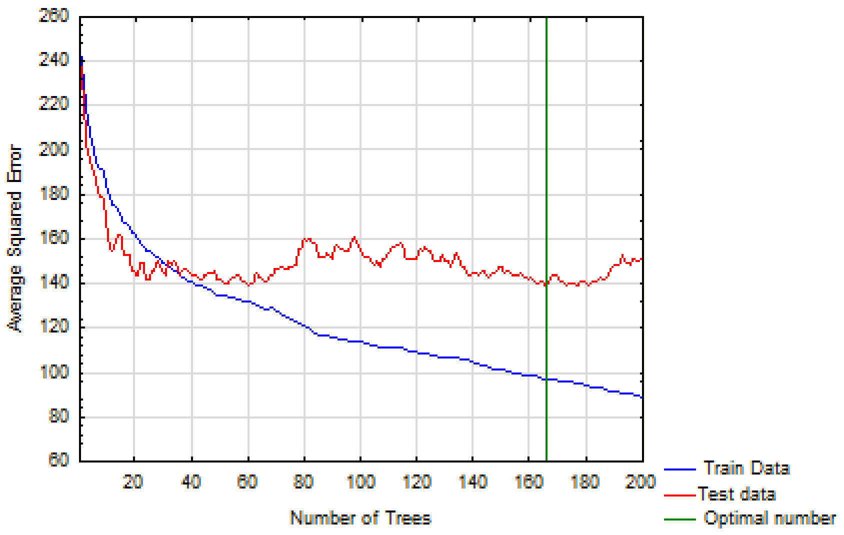

Figure 1. Optimization Graphic of Boosted Regression Tree Analysis for Programming Performance (Optimal number of trees, 166)

Figure 1 represents a plot of the prediction error function for the training data over successive trees and also an independently sampled testing data set at each stage. According to the results of the analysis, the eight variables together predict 65% of programming performance. In the determination of the prediction power of each variable separately in the boosted model, the relative importance weight and variable rank parameters have been analyzed. The importance weight and variable rank values of the variables that contribute to the prediction of the dependent variable are given in Table 3.

Table 3. Variable Ranks and Importance Weights Related to the Variables which Explain the Boosted Regression Tree Analysis

	Variables	Importance
Spatial Orientation	100	1
Spatial Memory	97	0.965*
Self-efficacy	93	0.929*
Mental Rotation	88	0.876*
Verbal Memory	83	0.830*
University	81	0.809*
Gender	37	0.365
Prior-knowledge	17	0.169

* = Importance level> 0.4 variables

Numbers in Table 3 indicated that, while the spatial orientation is the variable with the highest relative importance weight among the analyzed variables in relation to the prediction of programming performance (importance level= 1), spatial memory takes the second place (importance level = 0.965), self-efficacy perception the third place (importance level = 0.929), mental rotation skill takes fourth (importance level = 0.876) and these were followed by mental memory in fifth (importance level = 0.830) and the university variable in sixth place (importance level = 0.809). The variables of gender (importance level = 0.365) and prior knowledge (importance level = 0.169) took the last two places in order of importance in predicting the programming performance. This finding shows that cognitive skills, self-efficacy perception and university variables indicate almost equal importance in the prediction of programming performance, whereas the importance of gender and prior knowledge are relatively low. The scatter graph which shows the relationship between observed and predicted values in relation to programming performance is given in Figure 2.

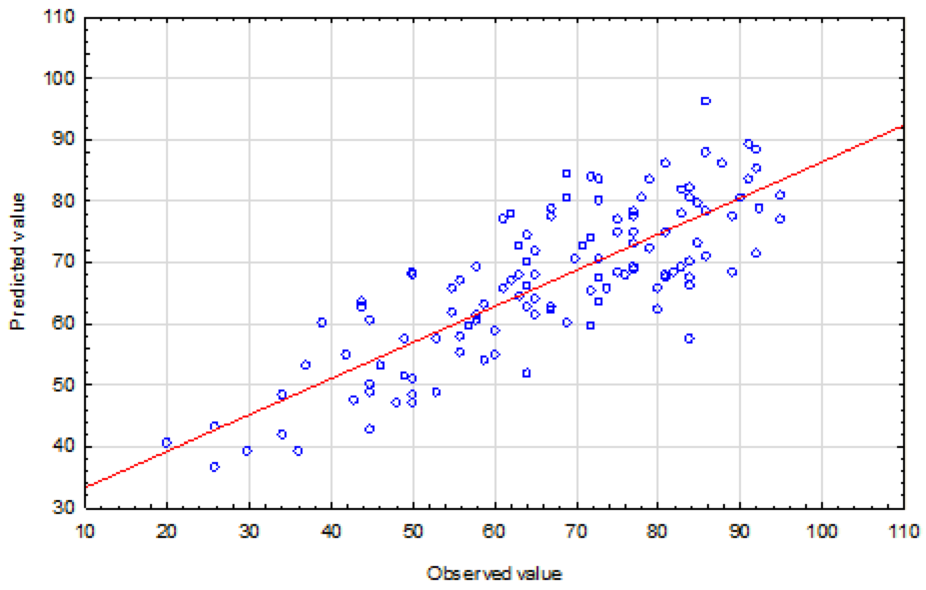

Figure 2. Correlation between Predicted and Observed Values for Programming Performance in Boosted Regression Tree Analysis

In order to verify the model created in relation to programming performance, Random Forests method was applied. For the random trees analysis, the total number of trees has been set to 500. The random test data proportion to be used for verification has been determined as 0.1 (10%) and the most suitable sub-sampling value has been determined as 0.56.

For stop parameters, the following values have been set: "maximum node number: 100, minimum number of observations possible in the child node: 5, minimum number to stop, which controls division in a manner where the minimum number indicated in all terminal nodes does not remain more than the observation five and the maximum level possible in each tree (the depth of the tree from the root node to the highest node): 10. The total number of trees included in the analysis is 500 and the maximum tree dimension is 100. In the analysis process, the average error square graphic which reflects the development process related to the consecutive average proportion on the test and education data is shown in Figure 3.

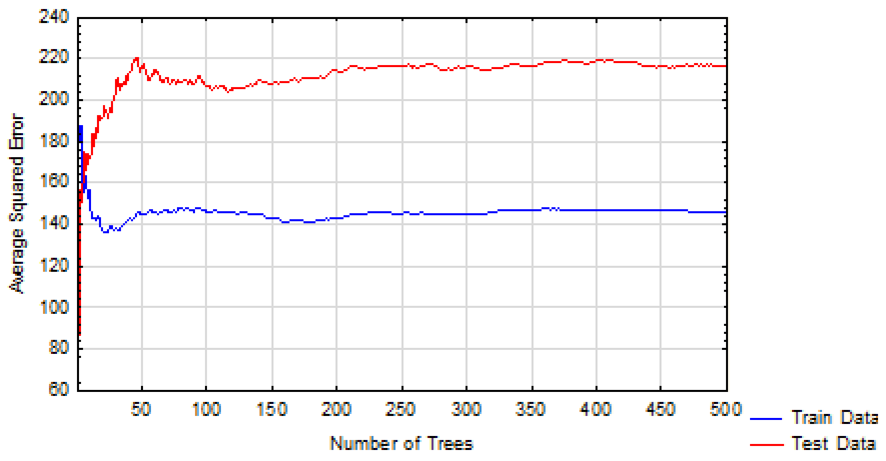

Figure 3. Average Error Squares Estimation Graphic for Programming Performance (Number of trees, set to 500)

Figure 3 reflects the basic operation in the over fitting prevention of the Random Forest algorithm. The most simple trees are added to this model, the more the erroneous classification related to the education set will decrease (StatSoft, 2013). According to the results of the Random Forests analysis, the eight variables (mental rotation, spatial orientation, visuo-spatial memory, verbal memory, self-efficacy perception related to programming, prior knowledge, the university students were attending, and gender) together predicted 60% of programming performance. In the identification of prediction power of each variable in the boosted model separately, the relative importance weights and variable rank parameters have been analyzed. The relative weight of importance of the variables and rank of the variables are given in Table 4.

Table 4. Variable Ranks and Their Weight of Importance in Relation to Random Forests Analysis Explanatory Variables

	Variable Rank	Importance
University	100	1*
Self-efficacy	95	0.952*
Mental Rotation	48	0.483*
Spatial Memory	47	0.474*
Spatial Orientation	47	0.470*
Verbal Memory	42	0.416*
Prior knowledge	22	0.215
Gender	19	0.186

* = Importance level > 0.4 variables

Table 4 presents the predictive values for programming performance, created with the Random Forests analysis. In this model, importance weight was set to 0.4 or over to estimate the predictive contribution significantly and the identified variables were determined as university, self-efficacy perception, mental rotation, spatial memory, spatial orientation and verbal memory. This finding completely overlaps with the finding achieved as a result of boosted regression analysis. However, the rank of importance of the variables in accordance with the boosted regression analysis becomes different. While the variable with the highest relative importance of weight is university, self-efficacy takes the second place, mental rotation takes the third, spatial memory takes the fourth place. These are followed by spatial orientation and verbal memory in the fifth and sixth places. Prior knowledge and gender as variables take the last two places in terms of their ranks of importance in predicting programming performance (See Table 4). The scatter graph shows the relationship between the observed and predicted values related to programming performance (See, Figure 4).

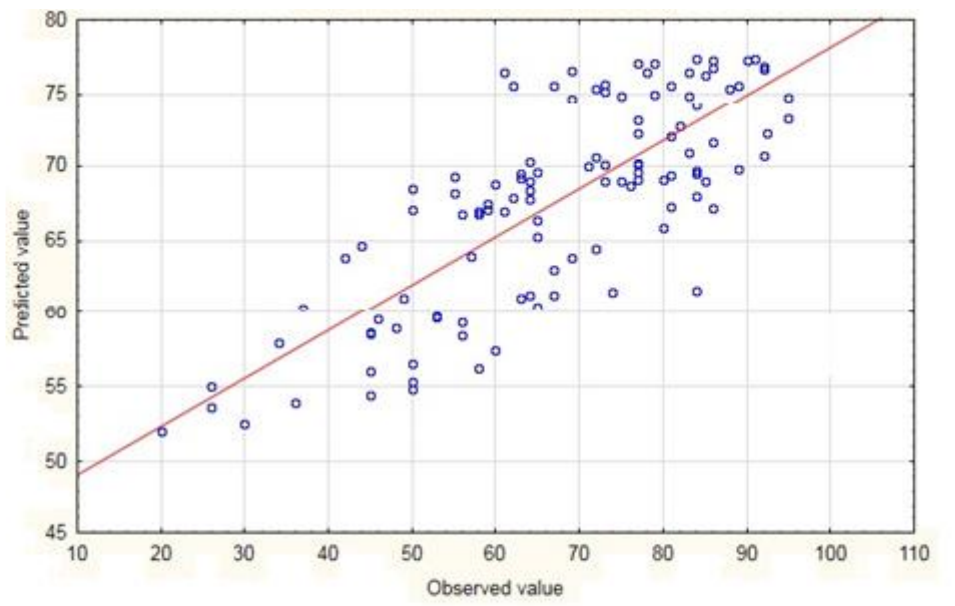

Figure 0. Correlation between Predicted and Observed Values for Programming Performance in Random Forests Analysis

The results of the Random Forests analysis have shown similar results to the boosted regression tree analysis. While the importance weight proportions in dependent variable prediction have been observed to be very close, the variables which contributed to the model have been observed to be the same yet in slightly different order. While cognitive skills, self-efficacy perception and the university students were attending have been identified as significant variables which contribute to the model, the importance weight of variables of gender and prior knowledge have been shown to be very low.

Conclusion and Discussion

This study explored the effects of various self-report and cognitive performance data on undergraduate students' computer programming performances from individual differences paradigm. The model included variables related to cognitive skills such as spatial skills (spatial orientation and mental rotation) and working memory (spatial memory and verbal memory), together with self-efficacy perception related to programming, gender, prior knowledge and the university students attend. At first, the correlation matrix had been evaluated, which indicated a very low ($r<0.5$, $p<0.05$) correlation between programming performance and self-efficacy perception and spatial memory and verbal memory; yet, no correlations between the other variables had been significant. This finding can be interpreted as the lack of collinearity among predictor variables, which implies that the order of variables entering the model may not affect the model leading to bias in relative importance among predictor variables.

First of all, the results showed that no linear relationship was observed among variables. This finding indicated that the interaction between the selected social cognitive skills of individuals and their performance (or success) could be dynamic; in other words, individuals who are good at particular skills could have been poor at other skills. In fact, Carello and Moreno (2005) also expressed that the interaction between the response times given by individuals in cognitive systems as generally being the common product of a series of the effects of components, and that these components will be complex and a linear assumption will limit us to predict this interaction. Luft et al. (2013) stated that the relationship between low and high performances and cognitive skills will not always be linear; therefore, educators are cautioned not to overgeneralize their beliefs when teaching computer programming.

According to the boosted regression tree analysis results that yielded the 65% of programming performance, the most important variable has been determined as the spatial orientation skill in terms of programming performance prediction. On the other hand, spatial memory, mental orientation, self-efficacy perception and verbal memory have been identified as variables which contributed to programming performance prediction the most. Yet, the effect of prior knowledge and gender on programming performance has not been found to be significant. This finding shows that individuals' cognitive skills have a very important effect on programming performance. In order to cross-validate the models, Random Forests has been applied to test it. According to the result of the Random Forests analysis, while 60% of programming performance has been predicted, the variables, which contributed significantly to the modeling of programming, remained the same and their rank of importance slightly changed. When the calculated results from both techniques are compared, the prediction rate for both techniques has been found to be similar and although the significant variables have not changed, the rate predicted with the boosted regression trees technique shows that this technique is acceptable in the interpretation of findings within the scope of the study. The variables, which contribute significantly to the prediction of the dependent variable in the model, may be taken as cross-validation of the other technique.

The study findings show that individuals' cognitive skills are the determinant variables in computer programming performances. This finding supports the existing studies which suggest that programming is an individual activity and thus, programming performance should be analyzed with a cognitive approach and individuals' cognitive skills and processes should be taken into consideration in teaching programming (Ambrosio et al., 2011; Lin et al., 2016). As Sterling and Brinthaupt (2003) once put forward, certain set of cognitive skills in particular is one of the most important success criteria within the context of rank of importance. Moreover, adaptive instructional strategies and media can be designed to help low performance students (Lin et al., 2016). In educational context, however, students acquire knowledge through a social-cognitive process, mainly by observing models, with a goal-directed behavior which may or may not be observable as a result of their internal process (Bandura, 1995). Furthermore, as social cognitive theory posits, learning most likely occurs if there is a close identification between the observer and the model and if the observer also has a good deal of self-efficacy. According to Bandura (1995), self-efficacy is "the belief in one's capabilities to organize and execute the courses of action required to manage prospective situations". Therefore, although technical skills are given importance in both training the programmers in schools and the selection of programmers in industry, other socio-cognitive variables, such as goal-directedness, modelling, and the mode of feedback could also be expanded in further research and be integrated as an instructional intervention.

The university students attended has emerged as an important variable which contributed to the prediction of their programming performance. Although the context of the introductory course to programming are similar in each university, this finding can be explained through various factors such as the number of instructors giving the course, evaluation of the course and grading of the exams. In his study, Mancy (2007) analyzed the relationship between working memory and programming performance and Mancy (2007) concluded that when the scoring methods are changed, the correlation coefficient between working memory and programming performance also changes. Therefore, since scoring method can change in different universities as different exam questions can be used, the university factor might have played a determinant role in predicting programming performance. In order to maintain a standard in learning outcomes, competency based evaluation can be implemented at schools.

Prior knowledge has not emerged as an important variable in modeling the programming performance in the study. In related literature, different findings exist. In some studies, prior knowledge emerges as a significant predictor of success related to programming (Wiedenbeck, 2005), whilst in others a relationship between having prior knowledge or not and programming performance was not found (Jegede, 2009). The reason for the different findings in relation to prior knowledge may be that data taken as a representation of prior knowledge is different for each study. In her study, which was conducted to determine factors that promote success in an introductory college computer science course, Wilson (2002) found that although programming experience (which included both a previous programming course and self-initiated programming) was not found to be significant in his full model, when the different types of computing experiences were compared as predictors some of them were significant. Jegede (2009) had dealt with prior knowledge in four different dimensions and he found that programming experience and the number of program writing years are not related to self-efficacy while the number of programming courses taken and success grades related to programming courses have a significant relationship with self-efficacy.

Gender was not an important variable in the prediction of programming performance. This shows that programming performances do not differ in accordance with whether the individuals are male or female. This finding is in parallel with many studies in related literature (e.g., Byrne and Lyons, 2001;) Milic (2009) and Pillay & Jugoo (2005) have shown that gender does not have a significant effect on students' programming performance. On the other hand, there are also contrary findings which suggest that gender has an important role in programming performance. While Lau and Yuen (2011) have put forth findings in favor of males in terms of programming performance, Yurdugul and Askar (2013) have observed in their study related to the development of programming knowledge that reach in males is higher and Aşkar and Davenport (2009) have also found out that self-efficacy perception related to programming in males is once again significantly higher in males. Beyer, DeKeuster, Walter, Colar and Holcomb (2005) have shown that gender differences are in question at the beginning of introductory courses on programming, however this difference decreases towards the end of the semester.

Lastly, some limitations of the study should be noted. First of all, this study is limited to the participation of 129 undergraduate students in three universities. Although these universities accept students from similar percentile based on a centralized university exam results; yet, students' developmental trajectories might vary during their two years of attendance. The instructors might also be another intervening effect, which leads to another limitation. Research with standardized testing on performance might overcome this issue. Another limitation could be attributed to the statistical procedures. Due to the small sample, one can argue 10 % of the participants for pre-training might not be enough for the selected sample size. More research is needed to validate or reject the findings of this study.

References

Ackerman, P. L., Beier, M. E., & Bowen, K. R. (2002). What we really know about our abilities and our knowledge. *Personality and Individual Differences, 33*, 587-605.

Altun, A., & Mazman, S. G. (2012). Programlamaya iliskin oz yeterlilik algisi olceginin Turkce formumun gecerlilik ve gvvenirlik calismasi [*Reliability and validity study on Turkish version of perceived self-efficacy scale for programming]. Egitimde ve Psikolojide Olcme ve Degerlendirme Dergisi, 3*(2), 297-308.

Alwin, D. F. (1994). Aging, personality and social change: The stability of individual differences over the adult life-span. In D. L. Featherman, R. M. Lerner & M. Perlmuter (Eds.), *Lifespan development and behavior*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Ambrósio, A. P., Costa, F. M., Almeida, L., Franco, A., & Macedo, J. (2011, October). Identifying cognitive abilities to improve CS1 outcome. In *Frontiers in Education Conference (FIE),* F3G-1.

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 - TOJET, 8*(1), 26-32.

Baddeley, A. (1992). Working memory. *Science, 255*(5044), 556-559.

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review, 84*, 191-215.

Bandura, A. (1995). Self-efficacy in changing societies. New York: Cambridge University Press.

Bergersen, G. R. & Gustafsson, J. E. (2011). Programming skill, knowledge, and working memory among professional software developers from an investment theory perspective. *Journal of Individual Differences, 32*(4), 201-209.

Bergin, S. & Reilly, R. (2005). Programming: Factors that Influence Success. *ACM SIGCSE Bulletin.,* 37(1), 411-415.

Bergin, S. & Reilly, R. (2006). Predicting introductory programming performance: A multi-institutional multivariate study. *Computer Science Education*, 16(4), 303-323.

Beyer, S., DeKeuster, M., Walter, K., Colar, M., & Holcomb, C. (2005). Changes in CS students' sttitudes towards CS over time: An examination of gender differences. *SIGCSE Bull., 37*(1), 392-396.

Blasko, D., Holliday-Darr, K., Mace, D., & Blasko-Drabik, H. (2004). VIZ: The visualization assessment and training Web site. *Behavior Research Methods, Instruments, & Computers, 36*(2), 256-260.

Blustein, J. & Satel, J. (2003). Spatial Ability and Information Shape: When do individual differences matter *Technical Report CS-2003-11*. Canada: Faculty of Computer Science Dalhousie University.

Breiman, L., (2001). Random forests. *Machine Learning*, 45, 5-32.

Breiman, L. & Cutler, A. (2004). Random forests. Retrived on 24 August 2016 from http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

Bruyer, R. & Brysbaert, M. (2011). Combining speed and accuracy in cognitive psychology: Is the inverse efficiency score (IES) a better dependent variable than the mean reaction time (RT) and the percentage of errors (PE)? *Psychologica Belgica, 51*(1), 5-13.

Byrne, P. & Lyons, G. (2001). The effect of student attributes on success in programming. *SIGCSE Bulletin, 33*(3), 49-52.

Buston, P. M. & Elith, J. (2011). Determinants of reproductive success in dominant pairs of clownfish: A boosted regression tree analysis. *Journal of Animal Ecology*, *80*(3), 528-538.

Carello, C. & Moreno, M. A. (2005). Why nonlinear methods. In M. A. Riley and G. C. van Orden (Eds.), *Tutorials in contemporary nonlinear methods for the behavioral sciences* (pp.1-25). Retrieved on 24 August 2016 from https://www.nsf.gov/sbe/bcs/pac/nmbs/nmbs.pdf

Caspersen, M. E. (2007). *Educating novices in the skills of programming* (Unpublished doctoral dissertation). Aarhus, Denmark: University of Aarhus, Department of Computer Science.

Charlton, J. P. & Birkett, P. E. (1999). An integrative model of factors related to computing course performance. *Journal of Educational Computing Research, 20*(3), 237-257.

Chen, C. (2000). Individual differences in a spatial-semantic virtual environment. *Journal of the American Society for Information Science, 51*(6), 529-542.

Cevik, V. (2012). *The roles of working memory capacity and instructional strategy teaching in complex cognitive task performances* (Unpublished doctoral dissertation). Ankara, Turkey: Hacettepe University Department of Computer Education and Instructional Technologies.

Daneman, M. & Merikle, P. M. (1996). Working memory and language comprehension: A meta-analysis. *Psychonomic Bulletin & Review, 3*(4), 422-433.

deRaadt, M., Hamilton, M., Lister, R., Tutty, J., Baker, B., Box, I., & Tolhurs, D. (2005, July). Approaches to learning in computer programming students and their effect on success. *Paper presented at the 28th HERDSA Annual Conference: Higher Education in a Changing World.* Sydney, Australia.

Elith, J., Leathwick, J. R., & Hastie, T. (2008). A working guide to boosted regression trees. *Journal of Animal Ecology*, *77*(4), 802-813.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367-378.

Genuer, R., Poggi, J. M., & Tuleau-Malot, C. (2010). Variable selection using random forests. *Pattern Recognition Letters*, *31*(14), 2225-2236.

Hannay, J. E., Arisholm, E., Engvik, H., & Sjøberg, D. I. (2010). Effects of personality on pair programming. *Software Engineering, IEEE transactions on education, 36*(1), 61-80.

Haavisto, M.-L., & Lehto, J. E. (2005). Fluid/spatial and crystallized intelligence in relation to domain-specific working memory: A latent-variable approach. *Learning and Individual Differences, 15*(1), 1-21.

Hoskinson, P. (2012). *Brain Workshop – a Dual N-Back game.* Retrieved on 20 December 2012 from http://brainworkshop.sourceforge.net/.

Howard, E. V. (2002). Can we teach introductory programming as a liberal education course? Yes, we can. *The Proceedings of ISECON* (Vol. 19). San Antonio, TX.

Irons, D. M. (1982). Cognitive correlates of programming tasks in novice programmers. *Proceedings of the 1982 Conference on Human Factors in Computing Systems*. Gaithersburg, Maryland, USA.

Jaeggi, S. M. , Buschkuehl, M., Perrig, W. J., & Meier, B. (2010). The concurrent validity of the N-back task as a working memory measure, *Memory*, *18*(4), 394-412,

Jegede, P. O. (2009). Predictors of Java programming self-efficacy among engineering students. *International Journal of Computer Science and Information Security, 4*(1-2). Retrieved on 24 August 2016 from https://arxiv.org/ftp/arxiv/papers/0909/0909.0074.pdf

Jenkins, T. (2002). *On the difficulty of learning to program.* Paper presented at the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences. Loughborough University, United Kingdom.

Jonassen, D. H. & Grabowski, B. L. (1993). *Handbook of individual differences learning and instruction*. London: Routledge.

Jones, S. J. & Burnett, G. (2008). Spatial ability and learning to program. *Human Technology, 4*(1), 47-61.

Kozhevnikov, M., & Hegarty, M. (2001). A dissociation between object manipulation spatial ability and spatial orientation ability. *Memory & Cognition, 29*(5), 745-756.

Lau, W. W. F., & Yuen, A. H. K. (2011). Modeling programming performance: Beyond the influence of learner characteristics. *Computers & Education, 571*(1), 1202-1213.

Lawton, C. (2010). Gender, spatial abilities, and wayfinding. In J. C. Chrisler & D. R. McCreary (Eds.), *Handbook of gender research in psychology* (pp. 317-341). New York: Springer.

Lehman, S., Bruning, R., & Horn, C. (1983). A tool for improving critical thinking in web-based instruction: Two experimental studies. *The CLASS project*. The Center for Instructional Innovation of the University of Nebraska.

Lin, Y. T., Wu, C. C., Hou, T. Y., Lin, Y. C., Yang, F. Y., & Chang, C. H. (2016). Tracking students' cognitive processes during program debugging—An eye-movement approach. *IEEE* transactions on education, *59*(3), 175-186.

Luft, C. D. B., Gomes, J. S., Priori, D., & Takase, E. (2013). Using online cognitive tasks to predict mathematics low school achievement. *Computers & Education, 67*(0), 219-228.

Mancy, R. & Reid, N. (2004). *Aspects of cognitive style and programming.* Paper presented at the 16th Workshop of the Psychology of Programming Interest Group (PPIG 16). Carlow, Ireland

Mancy, R. (2007). *Explicit and implicit learning: The case of computer programming* (Unpublished doctoral dissertation). University of Glasgow, United Kingdom.

Mason, R., Seton, C., & Cooper, G. (2016). Applying cognitive load theory to the redesign of a conventional database systems course. *Computer Science Education*, *26*(1), 68-87.

Mazman, S. G. & Altun, A. (2013). Individual differences in spatial orientation performances: An eye tracking study. *World Journal on Educational Technology, 5*(2), 266-280.

McGee, M. G. (1979). Human spatial abilities: Psychometric studies and environmental, genetic, hormonal, and neurological influences. *Psychological Bulletin, 86*(5), 889-918.

Merrienboer, J. J. G. V., & Paas, F. G. W. C. (1990). Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. *Computers in Human Behavior, 6*(3), 273-289.

Milic, J. (2009). Predictors of success in solving programming tasks. . *The Teaching of Mathematics, 12*(1), 25-31.

Pak, R., Rogers, W. A., & Fisk, A. D. (2006). Spatial ability subfactors and their influences on a computer-based information search task. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *48*(1), 154-165.

Pajares, F. (1996). Self-efficacy beliefs in academic settings. *Review of Educational Research, 66*(4), 543-578.

Pillay, N. & Jugoo, V. R. (2005). An investigation into student characteristics affecting novice programming performance. *SIGCSE Bulletin, 37*(4), 107-110.

Ramalingam, 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., Pérez-González, J. C., & Jiménez-Fernández, C. (2016). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior*, *72*, 678-691.

Rowan, T. C. (1957). Psychological tests and selection of computer programmers*. Journal of the ACM, 4(3*), 348-353.

Shute, V. J. (1991). Who is likely to acquire programming skills? *Journal of Educational Computing Research, 7*(1), 1-24.

Stalcup, K. A. A. (2005). *Multimedia learning: Cognitive individual differences and display design techniques predict transfer learning with multimedia learning modules.* (Unpublished doctoral dissertation). Graduate Faculty of Texas Tech University.

Sterling, G. D. & Brinthaupt, T. M. (2003). Faculty and industry conceptions of successful computer programmers. *Journal of Information Systems Education, 14*(4), 417-424.

Svedin, M. & Balter, O. (2016). Gender neutrality improved completion rate for all. *Computer Science Education*, 26(2-3), 192-207.

Vicente, K. J. & Williges, R. C. (1988). Accommodating individual differences in searching a hierarchical file system. *International Journal of Man-Machine Studies, 29*(6), 647-668.

Wagenmakers, E.-J., Maas, H. J., & Grasman, R. P. P. (2007). An EZ-diffusion model for response time and accuracy. *Psychonomic Bulletin & Review, 14*(1), 3-22.

Wiedenbeck, S. (2005). *Factors affecting the success of non-majors in learning to program.* Paper presented at the First International Workshop on Computing Education Research. Seattle, WA, USA.

Willman, S., Lindén, R., Kaila, E., Rajala, T., Laakso, M. J., & Salakoski, T. (2015). On study habits on an introductory course on programming. *Computer Science Education*, *25*(3), 276-291.

Wilson, B. C. (2002). A study of factors promoting success in computer science including gender differences. *Computer Science Education, 12*(1-2), 141-164.

Yurdugul, H. & Askar, P. (2013). Learning programming, problem solving and gender: A longitudinal study. *Procedia - Social and Behavioral Sciences, 83*(0), 605-610.

Wright, R., Thompson, W., Ganis, G., Newcombe, N., & Kosslyn, S. (2008). Training generalized spatial skills. *Psychonomic Bulletin & Review, 15*(4), 763-771.

Correspondence: Arif Altun, Professor, Department of Computer Education and Instructional Technologies, Faculty of Education, Hacettepe University, Beytepe, Ankara, Turkey.