Altun and Mazman *Smart Learning Environments* (2015) 2:13
DOI 10.1186/s40561-015-0020-0

Smart Learning Environments
a SpringerOpen Journal

RESEARCH Open Access

Identifying latent patterns in undergraduate Students' programming profiles

Arif Altun[1*] and Sacide Guzin Mazman[2]

* Correspondence: altunar@hacettepe.edu.tr
[1]Department of Computer Education and Instructional Technology, Hacettepe University, College of Education, Beytepe, Ankara, Turkey
Full list of author information is available at the end of the article

Abstract

This study aims to explore and reveal profiling patterns in the measurement of cognitive and noncognitivecharacteristics of undergraduate students' programming performances. Spatial skills, workingmemory, perceived programming self-efficacy, mathematics scores, and academic grade point averagescores were taken indicative variables to be explored. Participants of the study are 100 undergraduatestudents registered to the Programming-I course at two different universities. The data were analyzedthrough multi-dimensional profile analysis. The result of the multidimensional scaling analysis indicated twodifferent profiles for the two groups: high and low programming performance groups. For both groups,relationship between the most similar variables was found to be verbal memory, mathematics achievementand perceived programming self-efficacy. The results indicated that there was a relatively similarrelationship between visual-spatial memory and spatial orientation skills in the low-performance group,while mental rotation skill was significantly different than the other variables. It was noted that two profiles forhigh- and low-performance groups were quite different in terms of mental rotation skill. It was also found that spatial orientation, visual-spatial memory and mental rotation performances were all different from eachother, and from the other three variables in the group with high programming performance. The mostdefinitive variables for low- and high-performance groups were self-efficacy, verbal memory andmathematics achievement. This study revealed that only verbal memory was the determinant variable inboth groups for working memory.

Introduction

It is believed that programming skills naturally carry a number of other critical skills for learning and education (Howard, 2002), and that successful programmers are perceived by society as bright and successful individuals, generally in every area (Byrne & Lyons, 2001). Byrne and Lyons (2001) underlined a common understanding that today's society needs qualified and educated university graduates for the industry, and often, these qualified and educated people are comprised of individuals successful in programming. In parallel, Aşkar and Davenport (2009) pointed out that programming has particularly gained importance in computer-related areas, and that is why it has become a compulsory course.

Programming is considered as one of the most important complicated cognitive skills (Bergersen & Gustafsson, 2011). Programming performance is comprised of two different processes: programming knowledge, being language structure, definitions and certain algorithms, and programming skill, being strategies required to use this knowledge (Caspersen, 2007). Programming knowledge is required for programming, yet not enough. Students need to know, besides programming knowledge, a strategy for how to use this knowledge in the programming process (Caspersen, 2007). Individuals are expected to possess such basic skills as methodological thinking and signification, ability to use technology, process comprehension and logical thinking (Holvikivi, 2010; White & Sivitanides, 2002).

Programming covers complicated new information, strategic knowledge about this information as well as skills for applying this strategic knowledge (Robins et al. 2003). On the other hand, many students have difficulty in learning programming, and thus fail to do so (as cited in Ambrosio et al. 2011; Aşkar & Davenport, 2009; Caspersen, 2007; Dehnadi, 2009; Jenkins, 2002; Mancy & Reid, 2004). Lau and Yuen (2011) indicated one of the areas of study for programming as the determination of factors influencing programming achievement.

There are many different reasons for students to find programming course difficult and fail in the course. Learning styles, motivation and similar structures may affect programming performance of the learners (Jenkins, 2002). Some reasons for failure might be attributed to learners themselves, some to teaching methods. Some are, on the other hand, caused by attitudes, expectations or prior experiences of learners or teachers (Jenkins, 2002). A literature review by Yousoof et al. (2007) reports four fundamental reasons for failure in the programming process.

These are:

1) Complicated writing style and concepts in programming languages
2) Cognitive overload in the learning process
3) Teaching method mistakes
4) Conceptual errors

There are several studies in the literature about programming performance (Erdoğan et al. 2008; Jenkins, 2002; Milic, 2009). These studies deal with many variables affecting the programming performance including mathematics and science achievement, prerequisite knowledge, success/failure attribution, perceived programming self-efficacy, encouragement, comfort level, working style preference, prior experience of programming, prior experience of computers except for programming, intelligence, computer attitude, cognitive style, learning style, mother tongue, cognitive development, socio-economic status, creativity, problem-solving skill, and gender.

To conclude, existing research had investigated various variables to understand their effects on programming performances separately. Yet, what variables make individuals differ from each other is salient so far. Therefore, this study aims to deal with factors related to programming in order to reveal profile patterns of high- and low-performance groups on the basis of these individual differences. Consequently, by analyzing individual differences, learning environments can adapt to individual learners more meaningfully.

Brief definitions of these variables and their relationship with programming performance are provided below.

Working memory and programming performance

Working memory may be summarized as a structure divided between storing and information processing demands, consisting of a working space with a limited capacity (Baddeley, 2000). As in all learning processes, programming education needs the use of memory because programming process requires multi-operation of several cognitive activities. It is stated that working memory is a determinant for programming performance due to its high correlation with such skills as reasoning and general intelligence (Shute, 1991).

Shneiderman and Mayer (1979) conducted research on the impact of working memory on programming processes. When a programmer is given initial programming information, the programmer focuses on the statements given in the task, and the information is taken into the short term memory. Then the programmer reaches to the prior information in the long-term memory to complete this task in a successful manner. Finally, the information given about the task and taken to the short term memory, along with the existing relevant information in the long-term memory, are transferred to the working memory. Solutions are produced based on the problem situation given in the working memory.

Vainio and Sajaniemi (2007) stated that programmers can keep only a limited part of the program in the memory due to the limitation of the working memory capacity. Mancy and Reid (2004) produced findings of a possible relationship between working memory capacity and programming achievement. Pena et al. (1992) reached more concrete results, and revealed that capacity of working memory is an important predictor of programming achievement in the first stages of learning how to code. To support this finding, Shute's study in 1991 concluded that one needs to have a high working memory capacity in order to be a good programmer. Bergersen and Gustafsson (2011) found that working memory capacity improves programming knowledge, which improves programming skills; thus, working memory indirectly affects programming skill. A review of the above findings will show that working memory capacity has a significant, positive impact on programming performance.

Spatial skills and programming performance

White and Sivitanides (2002) stated that a learner's cognitive skills should be at a level required to learn a programming language. If cognitive characteristics of a learner are below the level required to learn a programming language, the learner may be disappointed, and above that level the learner may get bored. Similarly, Holvikivi (2010) indicated that cognitive capacity is one of the factors affecting programming achievement. However, Jones and Burnett (2008) claimed that few studies examined the relationship between programming skill and spatial skills. Jones and Burnett (2008) stated that spatial skill is one of those individual differences thought to be related to programming skill. Spatial skills are claimed to be a dimension of intelligence, and it is possible to find several skills intermingled in spatial skills. Halpern (2000) defines spatial skill as a cognitive characteristic to measure the skill for conceptualizing spatial relationship between objects. Mental rotation is a component of spatial skill, which is one of the most examined structures of spatial skills. Mental rotation may be defined as correct visualization of rotating two- and three-dimensional objects in the mind. Another component of spatial skills that may be related to programming achievement are mental models. Mental models may be defined as a clear visualization and abstraction of a program in the mind (Jones & Burnett, 2008). Fisher et

al. (2006) stated that mental model is an abstract representation of the program, consisting of various information emerged as a result of the source code during navigation.

Spatial skills are important for virtual environments, as well as for real environments. Therefore, it is claimed that spatial skills may be related to the understanding of relevant code pieces during the programming process (Cox et al. 2005). Jones and Burnett (2007) determined that individuals with higher spatial skills navigate differently than those with lower spatial skills, and stated that individuals with higher spatial skills establish better mental models due to better understanding of the space.

A review of the literature shows that there are significant and meaningful relationships between programming performance/education and spatial skills as well as visual-spatial skills and mental models, which are the sub-dimensions of the spatial skills (DeRaadt et al., 2005; Fincher et al., 2005; Jones & Burnett, 2008; Lau & Yuen, 2011; Mayer et al. 1986; Ramalingam et al. 2004). Fincher et al. (2005) found a significant yet low level of relationship between spatial skill and 'Introduction to Programming' course achievement. Other studies resulted in significant and positive impact of spatial skills (Jones & Burnett, 2008; Mayer et al. 1986) and visual-spatial skills (DeRaadt et al., 2005) on programming achievement. There are also findings showing that well-developed and well-structured mental models have a positive impact on programming performance (Lau & Yuen, 2011; Ramalingam et al. 2004).

Academic achievement, mathematics achievement and programming performance

Lau and Yuen (2009) stated that students' previous academic achievements have a positive and significant impact on their programming performance. Nonetheless, Byrne and Lyons (2001) pointed out that it is not possible to say individuals with higher academic achievement will be successful in programming, since there are many students who are successful in several courses may fail to achieve in the programming course. Programming is a complicated tasks, so being a successful programmer requires having extensive experience and skills. Such skills may include problem solving and mathematical skills (Jenkins, 2002). Similarly, Ambrosio et al. (2011) have also indicated that mathematical skills are among those skills required for programming. There are many findings in the literature considering mathematical background as a predictor of programming achievement (Bergin & Reilly, 2005, 2006; Wilson, 2002; Wilson & Shrock, 2001). Byrne and Lyons (2001) stated that there is a strong relationship between mathematics achievement and programming performance, yet pointed out to the necessity of further empirical studies to prove this relationship between these two structures.

Ambrosio et al. (2011) stated that students who easily learn programming may have mental models acquired through other courses such as mathematics and science. This supports the positive relationship between programming achievement and achievement in mathematics and other courses.

Programming self-efficacy and programming performance

Bandura (1977) defines self-efficacy as a learner's own opinion about one's organization of required actions to reach a specific target, and successfully fulfilling it. Pajares (1996) states that perceived self-efficacy plays an important intermediary role between an

individual's knowledge and actions because an individual's belief in own skills has an important effect on one's behavior.

Of students at the same cognitive level, those with higher self-efficacy have higher perseverance whereas those with lower self-efficacy overrating the situation making it more complicated than it actually is (Aşkar & Davenport, 2009; Davidsson et al. 2010). Therefore, since self-efficacy is associated with an individual's performance and belief in one's self, it is possible to improve one's self-efficacy to improve his performance (Ramalingam et al. 2004).

The studies exploring the relationship between programming self-efficacy and programming performances yield different and contradicting findings. Cegielski and Hall (2006) stated that perceived programming self-efficacy is a meaningful predictor of programming performance; on the contrary, Wilson (2002) found that programming self-efficacy has no significant impact on programming achievement. Jegede (2009) pointed out prior programming experience's effect on programming performance through perceived self-efficacy. This means perceived programming self-efficacy may act as an intermediary. Based on a review of literature, it is possible to assert that, in order to have a better understanding of the impact of perceived self-efficacy on programming skills, more studies are needed to examine the relationship between these two variables.

To conclude, the purpose of this study is to explore (1) how cognitive and non-cognitive factors play a role in understanding undergraduate students' programming performance, and (2) to reveal profiling patterns in the measurement of cognitive and non-cognitive characteristics of undergraduate students according to their high and low programming performances.

Method

Profile analysis with multidimensional scaling is appropriate for analysis of sample of any sizes and includes each profile level and pattern information. Therefore, it is stated to be more advantageous than other profile analysis methods like cluster analysis and modal profile analysis (Kim et al., 2004). Therefore, in this study, ALSCAL procedure of multidimensional scaling method was used to describe profile differences of low and high programming performance groups across their cognitive and non-cognitive characteristics.

Participants

The participants included 100 sophomores who were enrolled in "Programming Language-I" course at the Computer Education and Instructional Technologies Department in two different universities. There were 54 (54 %) female and 46 (46 %) male participants in the study.

Measurements

Cognitive profile data were collected with computerized spatial orientation test, mental rotation test, visual-spatial working memory test and verbal working memory test. Programming self-efficacy scale was distributed in a single session at each department

separately in a paper-and-pencil form. Participants' programming performance scores, their GPA scores and their math scores were gathered from their course instructors.

Spatial orientation ability was measured with *Spatial Orientation Test* which was developed originally by Kozhevnikov and Hegarty (2001) and was standardized for computerized version through E-Prime 2.0 software by Mazman and Altun (2013) for Turkish undergraduates. This test consisted of 14 questions and both the reaction time and accuracy scores were logged for each question. In this test, participants see an array of objects on the screen. In this array, there have been eight objects (chair, car, cat, armchair, traffic light, dog, stop sign and bus) which were arranged like around a circle. A character head has placed at the center of objects. The character's eyes were looking towards the cat which is located at the right side of the array in each question. Participants were asked to imagine themselves standing at the characters' head and then move at the clockwise to one of the objects stated in the instruction and when the facing object is straight ahead, to indicate the direction of a third object of location on the answer section. In each trial, participants were to imagine themselves standing at the character's head, then face at clockwise to one of the figure specified by instruction and when the facing figure is straight ahead, indicate the direction to a third figure of location on the answer section.

Answer key consists of eight empty checkboxes which were placed at the point of eight basic directions (0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, 315^0) and aligned like forming a circle. The character's head was drawn at the center of checkboxes and the object which is imagined to being faced was drawn vertically up, being pointed by an arrow from center. (For more information about the test, see Mazman and Altun, 2013).

A revised version of visual spatial memory-word rotation test was used which was originally developed by Blasko et al. (2004) within the scope of "Visualization Assessment and Training Project (VIZ)" to asses, examine and improve spatial performance. Spatial memory-word rotation test was modified and revised by researchers through E-Prime 2.0 software and norm study was conducted for Turkish undergraduates. The test is consisted of three sets and difficulty level increases in each set. There have been two main subtasks in test; the first is mental rotation of numbers and the second is retention of peak direction of numbers in each trial. In each set, using one of the "2,7,1,4" numbers which have obvious peak, mirror or same image of one of the numbers is given on the screen for each question. Numbers appears as their peak direction shows one the eight different orientations (45^0, 90^0, 135^0, 180^0, 225^0, 270^0, 315^0, 360^0). The first participants were required to decide if the image of the number is a mirror view or normal view and then press the "M" from the keyboard if the view is mirror, or press the "N" if the view is normal. This section is where the participants performed mental rotation tasks.

While they were requested to decide whether the number image is in mirror or normal view, participants were also asked to remember the direction of each number peak. After finishing mirror/normal decision part in related set, they must select the checkboxes that show direction of numbers' peak without considering order. The numbers of directions which participants must remember increase one in every set. Each of the reaction time (ms) and accuracy score were logged in both part of the (mental rotation-visual spatial memory) question.

An n-back task software developed by Brain Workshop was used to measure *verbal working memory*. This software is developed by Hoskinson (2012) with Python

programming language and adapted into Turkish by Çevik and Altun (2012) compiling interface files. The test starts with Dual 2-Back mode by default. Dual part of the name implies the software presentation modality (sound, position, color, shape … etc.) and n-back implies the how many trial back participant is being asked to remember. Screen design consisted of a 3X3 matrix independently of modality. Since the software was used to measure only verbal working memory for this study, only auditory modality was considered and measurement was made in 2-back mode.

For the practice session at first auditory modality (10+4 trial), after that position modality (10+4 trial) and lastly synchronous auditory-position modality (15+4 trial) was executed. For the test session totally 30+4 trial auditory modality was set.

In auditory modality participants took a headset. A series of letters were presented auditory at the rate of 2.5 s per stimuli. Participants press "L" if the letter they hear is the same it was 2 trial back. For example, if the presented letters were arranged as "T K N M N H M", participants were expected to press "L" just they heard the second N. Because there is M between two N and the letter N is same with two back stimuli. Test scores and information details about the test session were logged into a txt file. The test scores were produced as a numerical score out of 100.

Participants' programming self-efficacy as a non-cognitive variable was measured with a self-report tool, "*Programming Self Efficacy Scale*" which was adapted into Turkish by Mazman and Altun (2013). The scale was 7-likert type with 9 items under two factors ("ability to perform simple programming tasks" and "ability to perform complex programming tasks"). The scale ranged from 1= not at all confident, 2= Mostly not confident, 3= slightly confident, 4= %50/50, 5= Fairly confident, 6= mostly confident, 7= absolutely confident. Maximum score was 63 and the minimum was 1. The Cronbach alpha coefficient was reported to be .928. The construct validity study of the scale yielded two factors, which have explained 80,814 % of the variance together.

For the *programming performance*, end of the year degree scores of Programming-1 course was gathered. For the math scores arithmetic mean of end of the year degree scores of Math I and Math II courses were considered. All the degree scores were recorded in the scale of 100. Programming-1 course was designed to introduce basic programming concepts and coding to undergraduate students. The structure of the course was as follows: First, students were instructed on reading the predefined codes to understand what outcomes will be produced when computation was executed. Secondly, they were taught variables, functions, methods, and events to be able to write codes for predefined problem sets. Finally, they were trained for error debugging regarding the codes they had practiced.

Data scoring – defining groups

The dependent variable of the study is the overall programming performance. The independent variables included spatial orientation ability, visual-spatial working memory, mental rotation ability, verbal working memory, academic grade and mathematic grades. Since visual spatial working memory, spatial orientation and mental rotation tests yielded two different measures, accuracy and reaction times, inverse efficiency scores which was proposed by Townsend and Ashby (1978) was applied. This score was calculated according to the following formula;

$$IES = \frac{\mathrm{RT}}{\mathrm{PC}}$$

RT= Mean reaction time of the correct responses

PC= Proportion of the correct responses

IES= Inverse Efficiency Score

IES was calculated for each participant in their spatial orientation, mental rotation and visual spatial memory test scores. Finally, all the programming performance scores were converted to Z scores to determine the high and low profiles. Participants that have positive z scores, which means above average, were assigned to high group and participant with negative z scores were assigned to low group. So, 45 of the participants were grouped in low programming profile (X=58,4, sd=9,12) and 55 of the participants were placed in high profile group (X=81,68; sd=6,8).

Data analysis

Data analysis was a two-step process. First, descriptive statistics of all the test scores were calculated. Then, participants were grouped according to their programming performances. Z scores of programming performance were calculated and participants were grouped as high and low programming profiles. Participants having positive z score were placed in high group while participants having negative z score were coded as low profile group. As a result, 45 of the participants were categorized under low group (X=58.4; sd=9.12) and 55 of the participants were categorized under high group (X= 81.68; sd=6.8).

Secondly, ALSCAL algorithm of the multidimensional scaling technique was run to explore the patterns of cognitive abilities in programming performance groups. Multidimensional scaling is defined as "…displaying the objects in a k-dimensional space through distances between an object determined by p variables. It is a graphical based method to show structure of relationship between objects" (Bağ and Alpar, 2011). For this purpose, distances between objects are reported as a coordinate which is represented in space. Those obtained coordinate data were visualized through MS Excel program.

While evaluating the goodness of multidimensional scaling models, stress value and the RSQ are two fit indexes that are used (Bağ and Alpar, 2011). Stress value is given by the differences in data and predicted value, and RSQ criteria is a fit index that shows the squared correlation between Euclidean distances and the disparities (Young, 2013).

Table 1 Descriptive statistics of participants' test scores

	N	$\bar{X}$	Sd	Min.	Max.
Programming Performance	100	71,2	14,07	34	96
IES_Spatial Orientation	100	1417,09 ms	1259,47	652,22	12153,46
IES_Mental Rotation	100	4450,68 ms	1736,93	1776,26	11531,28
IES_ visual Spatial Memory	100	1380,0313 ms	923,51	435,81	6729,73
Verbal Memory	100	59,33	19,22	13	100
Mathematic Performance	100	61,0545	22,36	4,5	97,50
Programming Self Efficacy	100	42	12,3	9	63

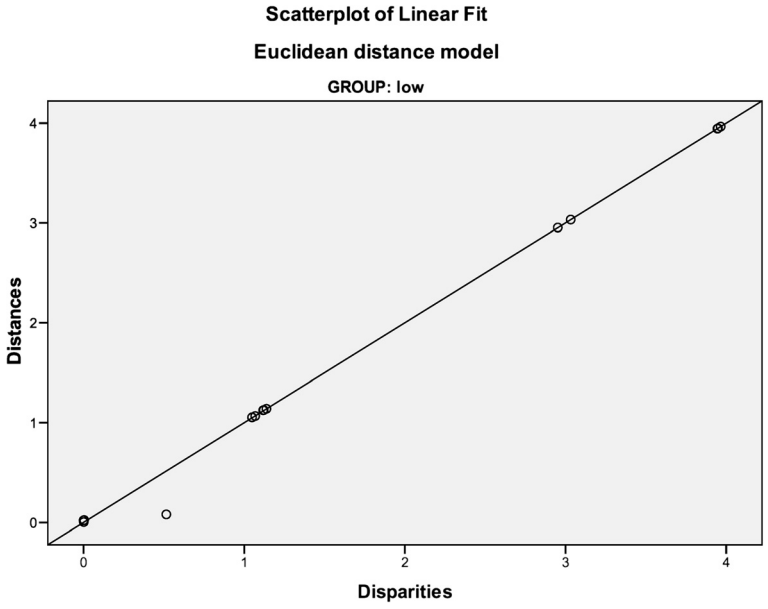

Fig. 1 Variable representation in Euclidian distance model for low profile group

Higher values of stress indicate poorer fit (0–0.025: excellent, 0.025-0.05: good, 0.05-0.1: fair, 0.1-0.2: poor).

Findings

As seen in Table 1, the mean score for the programming performance of a total of 100 participants were found to be 71.2 (sd=14.07) out of 100. Mean of mathematic performance was found to be 2.57 (sd=0.42) out of 4, while programming self-efficacy score was

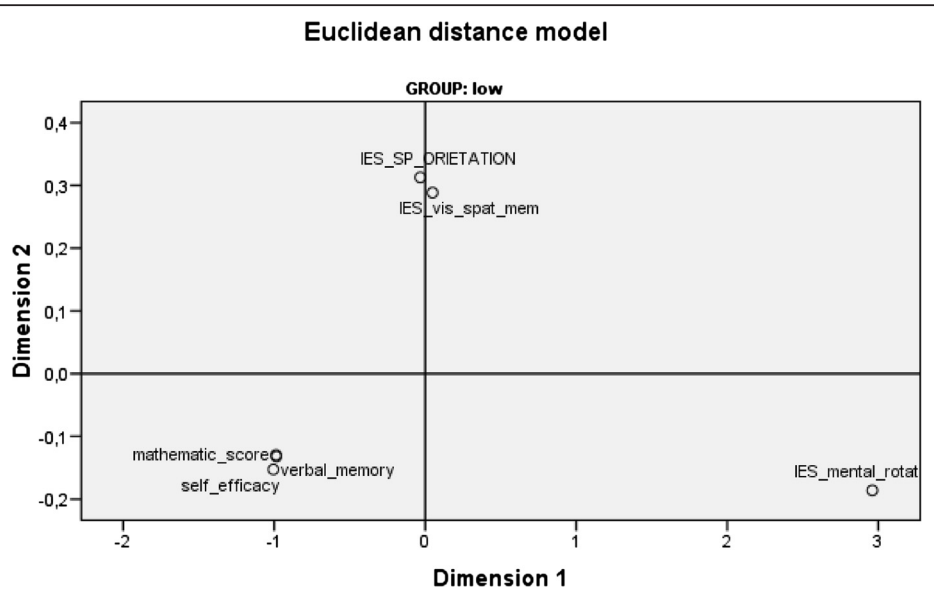

Fig. 2 Distances and similarities among profiles of low performance group members on a two-dimensional space

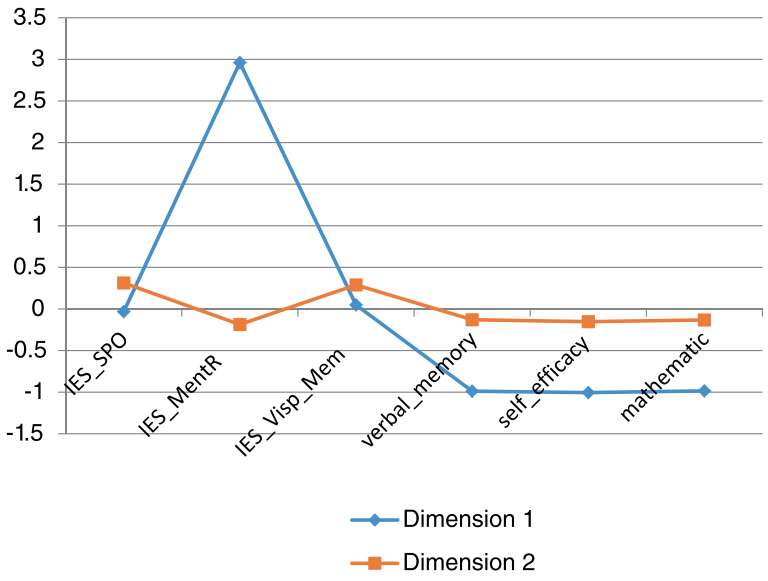

Fig. 3 Profile patterns of low performance group on dimension 1 vs dimension 2

found to be 42 (sd=12.3) out of 63. Descriptive statistics about the participants were presented in Table 1.

Patterns in high and low performance groups

Based on the analyses, two dimensional model was chosen for the model fit and s-stress was set to be .001. ALSCAL multidimensional scaling analysis yielded the model consisting of mental rotation, spatial orientation, visual-spatial working memory, verbal memory, self-efficacy, and math performance variables to be the best fitted with two-dimensional solution. Findings will be presented for low and high group separately belxow.

Profiles in low performance group

The analysis of goodness of fit indexes for low level group was found to be S-Stress=0.05 and RSQ=0.994. Those obtained goodness of fit index values showed that the model was in a good fit. The findings indicated that distances and differences among variables exhibit a linear relationship, which is presented in Fig. 1.

In order to explore how the variables are represented on a two-dimensional space, the distances and similarities among profile patterns of low performance group members are visualized in Fig. 2.

Based on the analysis results, it is seen that self-efficacy, verbal memory and mathematics scores get grouped together, while distance between visual -spatial memory and spatial orientation ability shows relative closeness, whereas mental rotation scores seemed to behave independently. This finding may be interpreted as low performance group members' perceived self-efficacy, mathematics achievement and verbal memories have a matching tendency. Their visual-spatial memory and spatial orientation skills are also relatively alike while their mental rotation skills differ significantly.

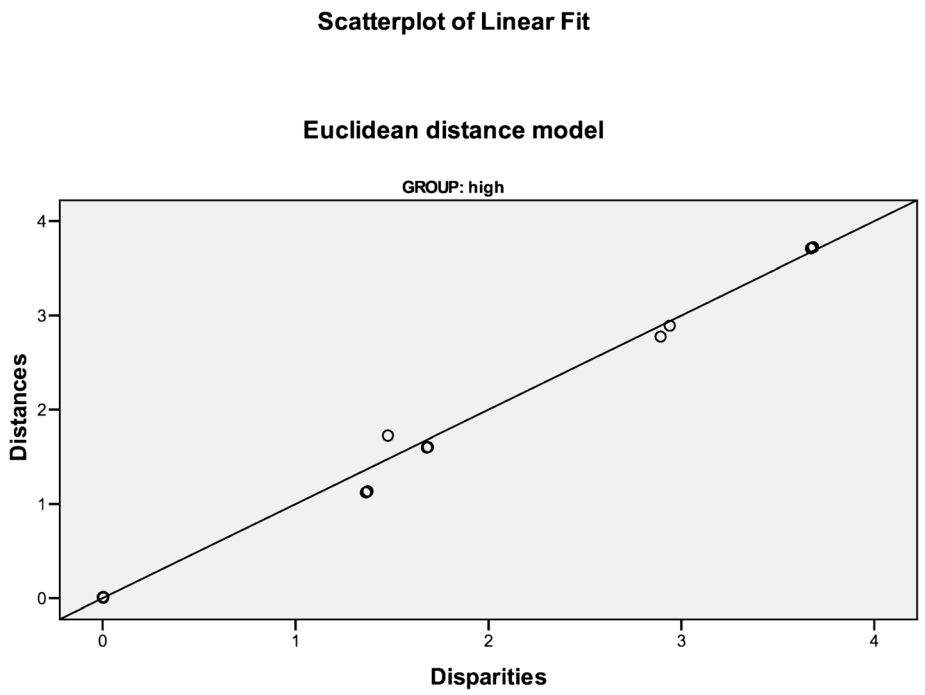

Fig. 4 Variable representation in Euclidian distance model for high profile group

Plot graphics of the coordinates obtained on the basis of participant profiles were drawn using MS Excel. Figure 3 displays the graphics for two profiles exhibited by the participants in the low performance group.

As seen in Fig. 3, although mental rotation skill is low both at the first and second dimension, it represents different patterns; besides, spatial orientation and visual -spatial memory are at a relatively similar level on both of the low level profile. Along these lines, a similar pattern was observed between verbal memory, perceived self-efficacy and mathematics achievement.

Profiles in high performance group

As a result of the multi-dimensional scaling analysis for the high programming performance group, iteration was continued until the number of dimensions was two, and the stress value was smaller than 0.001; and the iteration was stopped at the fourth iteration where the stress value reached to 0.00008. The analysis resulted in that the two-dimension models' goodness of fit indexes were found to be as S-Stress=0.060 and RSQ=0.990 for the high performance group.

Obtained fit index values indicated a good fit by the model. Figure 4 shows that distances and differences among variables indicate a linear relationship.

Figure 5 gives a demonstration of distances among patterns of high performance group members on a two-dimensional space.

Based on the analysis results, it is observed that self-efficacy, verbal memory and mathematics scores go together, while visual -spatial memory, orientation and mental rotation scores were independent. This finding may be interpreted as high performance group members' perceived self-efficacy, mathematics achievement and verbal memories have a

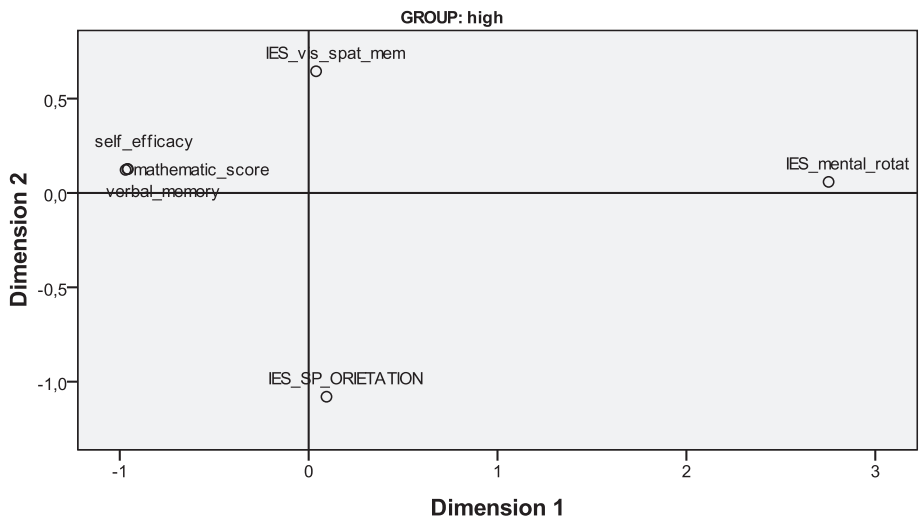

Fig. 5 Distances and similarities among profiles of high performance group members on a two-dimensional space

matching tendency, while their spatial orientation skills, visual -spatial memory and mental rotation skills significantly differ, which indicates that this group, with their higher programming performance, does not exhibit a corresponding distribution in itself in the latter skills.

Plot graphics of the coordinates obtained on the basis of participant profiles were presented in Fig. 6 for two profiles exhibited by the participants in the higher performance group.

There were two different patterns observed in exploring the profiles of high performance group. In the first dimension, there were a group of participants with a higher level of mental rotation, whereas spatial orientation and visual -spatial memory are at a relatively lower level. Verbal memory, perceived self-efficacy and mathematics achievement were observed at similar across high performance students, yet in a negative manner.

At the second dimension profile, spatial orientation skills exist at a higher score interval compared to the other group, while mental rotation is at a lower interval. The groups also act differently in terms of visual-spatial memory scores. On the other hand, verbal working memory, perceived self-efficacy and mathematics achievement are positive at a corresponding level. As a result, similar characteristics have been observed for verbal working memory, perceived self-efficacy and mathematics achievement both for first and for second dimension profiles. Yet, the two profiles differ in terms of mental rotation skill, spatial orientation skill and visual-spatial memory scores.

Consequently, in the light of the findings, it may not be sufficient to consider mental rotation skill for the low performance group on its own in order to model this group's performance. Similarly, since mental rotation, spatial orientation skill and visual -spatial memory act independently for the high performance group, use of these variables in the modelling process may provide different results. Instead, it would be better to consider verbal working memory, perceived self-efficacy and mathematics achievement as observed correspondingly for both groups to be noticeable.

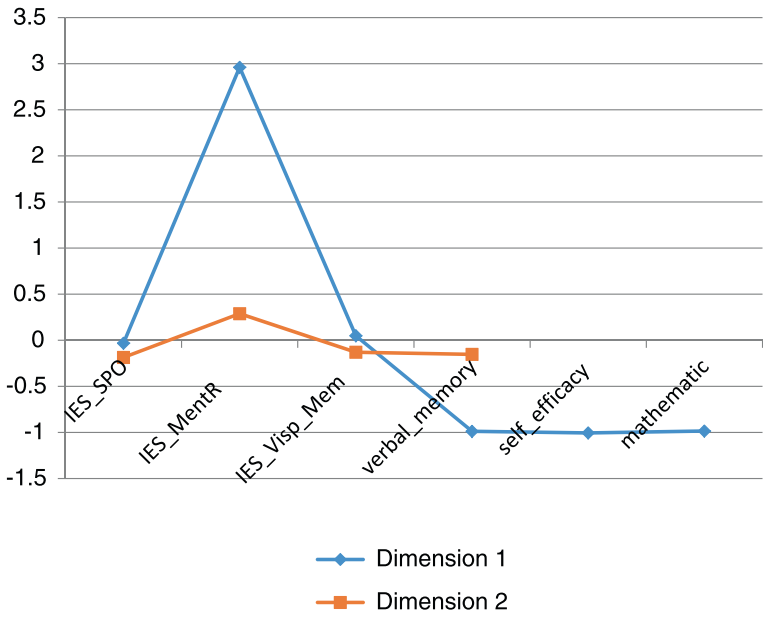

Fig. 6 Profile Patterns of High Performance Group on Dimension 1 vs Dimension 2

Conclusion and Discussion

This study aimed at exploring how various cognitive and non-cognitive variables had an impact on undergraduate students' programming performances. The data were analyzed to examine the patterns via multi-dimensional scaling analysis for two groups with higher and lower programming performance. Participants' programming performance were computed on the basis of calculated z scores, using end-of-term pass grades from the Programming Languages-I course, and those with negative z scores were classified as being in the low performance group, whilst those with positive z scores as being in the high performance group.

Results of the multi-dimensional scaling analysis of the variables associated with programming performance indicated two different profiles for two groups having both high and low programming performance scores. For both groups, a relationship between the most similar variables was found for verbal memory, mathematics achievement and perceived programming self-efficacy. In the low programming performance group, the relationship between visual-spatial memory and spatial orientation skills were relatively similar, while mental rotation skill was significantly different from all other variables. It is remarkable that two profiles emerged both for the high- and low-performance groups that are quite different in terms of mental rotation skill. In the high performance group, inversely, spatial orientation, visual-spatial memory and mental rotation performances are significantly different both from each other and from the other three variables.

The most determinant variables for the high- and low-performance groups are self-efficacy, verbal memory and mathematics achievement. The literature emphasizes the importance of working memory in providing both declarative information and practical skills in teaching programming (Shute, 1991); however, this study has concluded that only verbal memory was the determinant variable in both groups in terms of working memory. Interviews held with instructors of the Programming-I course and a review of

the exam questions in these courses show that the courses typically focus on the conceptual dimension of the programming process, and classes are realized through lecture-based mode mainly on syntax and procedures, followed by paper-and-pencil test. Therefore, it can be speculated that these test scores might have significant impact on bringing the verbal memory scores to the foreground.

On the other hand, the literature emphasizes the importance of spatial skills at different stages of the programming process, such as understanding the program and debugging, as well as navigating among code blocks in visualization of the program in the mind prior to operation (Vainio & Sajaniemi, 2007; Cox et al. 2005). However, this study concludes that other variables are quite independent for both groups, and are not determinant in profiles for high- and low- performance groups. This finding may be interpreted by the non-existence of spatial skills as a determinant variable since the subject course (Programming-I) is an introductory course and does not cover advanced programming performance processes, such as navigation between debugging and coding, and advanced algorithms.

This study examined programming performance according to the course scores, and thus it is limited to the structure of the exams. Besides, even though one group's programming performance is considered 'high', it is important to note that this is limited to the 2nd year 1st term final grades. It would be possible to obtain more comprehensive results with a similar study examining professional programmers as well, in order to see in-group and between-group differences. More research would shed a broader light upon our understanding of the programming performance as well as computational thinking. Secondly, only perceived self-efficacy scores were obtained as a non-cognitive factor in this study. More exploratory research is needed to explore other non-cognitive factors such as emotional stances. By taking individual differences into account, instructional designers could provide more meaningful design choices and smart learning environments for learners could be developed.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
AA designed the research with the variables and measurement tools. SGM carried out data collection process and analyses. Both authors contributed equally on reporting the findings and writing the final report.

Author details
[1]Department of Computer Education and Instructional Technology, Hacettepe University, College of Education, Beytepe, Ankara, Turkey. [2]Department of Computer Education and Instructional Technology, Uşak University, College of Education, Uşak, Turkey.

Received: 15 June 2015 Accepted: 26 August 2015
Published online: 17 September 2015

References

AP Ambrosio, FM Costa, L Almeida, A Franco, J Macedo, *Identifying cognitive abilities to improve CS1 outcome*. Paper presented at the Frontiers in Education Conference (FIE), Rapid City, SD, USA, 2011)
P Aşkar, D Davenport, An Investigation of Factors Related to Self-Efficacy for Java Programming Among Engineering Students. Turk. Online. J. Educ. Technol. - TOJET, **8**(1), 26–32 (2009)
AD Baddeley, The episodic buffer: a new component of working memory? Trends Cogn. Sci. **4**, 417–423 (2000)
HG Bağ, R Alpar, Multidimensional Scaling in Multivariate Statistical Methods, ed. by R Alpar (2011), pp. 383–404. Detay Publishing, Ankara, Turkey.
A Bandura, Self-efficacy: toward a unifying theory of behavioral change. Psychol. Rev. **84**(2), 191–215 (1977)
GR Bergersen, J-E Gustafsson, Programming Skill, Knowledge, and Working Memory Among Professional Software Developers from an Investment Theory Perspective. J. Individ. Differ. **32**(4), 201–209 (2011)
S Bergin, R Reilly, Programming: Factors that Influence Success. ACM SIGCSE. Bull. **37**(1), 411–415 (2005)
S Bergin, R Reilly, Predicting introductory programming performance: a multi-institutional multivariate study. Comput. Sci. Educ. **16**(4), 303–323 (2006)

DG Blasko, K Holliday-Darr, D Mace, H Blasko-Drabik, VIZ: The visualization assessment and training website. Behav. Res. Methods Instrum. Comput. **36**(2), 256–260 (2004)
P Byrne, G Lyons, The Effect of Student Attributes on Success in Programming. SIGCSE. Bull. **33**(3), 49–52 (2001)
ME Caspersen, Educating Novices in the Skills of Programming ((Unpublished PhD Dissertation), University of Aarhus Denmark, 2007)
CG Cegielski, DJ Hall, What makes a good programmer? Commun. ACM **49**(10), 73–75 (2006)
V Çevik, A Altun, Measuring Working Memory by Using a Multimedia Based Task. Hacettepe. Univ. J. Educ., Special Issue **1**, 32–40 (2012)
A Cox, M Fisher, P O'Brien, *Theoretical Considerations on Navigating Codespace with Spatial Cognition* (Paper presented at the 17th Workshop of the Psychology of Programming Interest Group, Sussex University, UK, 2005)
K Davidsson, L-Å Larzon, K Ljunggren, Self-Efficacy in Programming among STS Students. Reports. Uppsala University, Department of Information Technology (2010). Retrieved from http://www.it.uu.se/edu/course/homepage/datadidaktik/ht10/reports/Self-Efficacy.pdf on 21.11.2013
S Dehnadi, A cognitive study of learning to program in introductory programming courses ((Unpublished PhD Dissertation), Middlesex University, UK, 2009)
M DeRaadt, M Hamilton, R Lister, J Tutty, B Baker, I Box, D Tolhurs, *Approaches to learning in computer programming students and their effect on success*. Paper presented at the 28th HERDSA Annual Conference: Higher Eduation in a Changing World, The University of Sydney, Australia, 2005
Y Erdoğan, E Aydın, T Kabaca, Exploring the psychological predictors of programming achievement. J. Instr. Psychol. **35**(3), 264–270 (2008)
S Fincher, B Baker, I Box, Q Cutts, M Raadt, P Haden, J Tutty, Programmed to succeed? A multi-national, multi-institutional study of introductory programming courses. Technical Report (Computing Laboratory: University of Kent, UK, 2005)
M Fisher, A Cox, Z Lin, *Using Sex Differences to Link Spatial Cognition and Program Comprehension*. Paper presented at the 22nd IEEE International Conference on Software Maintenance (ICSM '06), Philadelphia, Pennsylvania, USA, 2006
DF Halpern, *Sex differences in cognitive abilities* (Lawrence Erlbaum Associates, Mahwah, NJ, USA, 2000)
J Holvikivi, Conditions for Successful Learning of Programming Skills, in *Key Competencies in the Knowledge Society*, ed. by N Reynolds, M Turcsányi-Szabó, vol. 324 (Springer, Berlin Heidelberg, 2010), pp. 155–164
P Hoskinson, Brain workshop - a dual n-back game. Retrieved 20 February, 2012 (2012). From: http://brainworkshop.sourceforge.net/
EV Howard, Can We Teach Introductory Programming as a Liberal Education Course? Yes, We Can! in *The Proceedings of ISECON 2002, v 19 (San Antonio)*, 2002. ISSN: 1542–7382
PO Jegede, Predictors of Java Programming Self Efficacy Among Engineering Students. Int. J. Comput. Sci. Inf. Secur. **4**, 1–2 (2009)
T Jenkins, *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, 2002)
SJ Jones, GE Burnett, Spatial skills and navigation of source code. SIGCSE. Bull. **39**(3), 231–235 (2007). doi:10.1145/1269900.1268852
SJ Jones, G Burnett, Spatial Ability and Learning to Program. Hum. Technol. **4**(1), 47–61 (2008)
S Kim, CL Frisby, ML Davison, Estimating cognitive profiles using profile analysis via multidimensional scaling (PAMS). Multivariate Behavioral Research **39**(4), 595–624 (2004)
M Kozhevnikov, M Hegarty, A dissociation between object manipulation spatial ability and spatial orientation ability. Mem. Cogn. **29**(5), 745–756 (2001)
WWF Lau, AHK Yuen, Exploring the effects of gender and learning styles on computer programming performance: implications for programming pedagogy. Br. J. Educ. Technol. **40**(4), 696–712 (2009)
WWF Lau, AHK Yuen, Modeling programming performance: Beyond the influence of learner characteristics. Comput. Educ. **57**(1), 1202–1213 (2011)
R Mancy, N Reid, *Aspects of Cognitive Style and Programming*. Paper presented at the 16 th Workshop of the Psychology of Programming Interest Group (PPIG 16) (Carlow, Ireland, 2004)
RE Mayer, JL Dyck, W Vilberg, Learning to program and learning to think: what's the connection? Commun. ACM **29**(7), 605–610 (1986). doi:10.1145/6138.6142
G Mazman, A Altun, Individual Differences in Spatial Orientation Performances: An Eye Tracking Study. World. J. Educ. Technol. **5**(2), 266–280 (2013)
J Milic, Predictors of success in solving programming tasks. Teach. Math. **12**(1), 25–31 (2009)
F Pajares, Self-Efficacy Beliefs in Academic Settings. Rev. Educ. Res. **66**(4), 543–578 (1996)
CM Pena, CM Pena, WC Tirre, Cognitive factors involved in the first stage of programming skill acquisition. Learn. Individ. Differ. **4**(4), 311–334 (1992). http://dx.doi.org/10.1016/1041-6080(92)90017-9
V Ramalingam, D LaBelle, S Wiedenbeck, *Self-efficacy and mental models in learning to program*. (Paper presented at the Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, Leeds, United Kingdom, 2004)
A Robins, J Rountree, N Rountree, Learning and Teaching Programming: A Review and Discussion. Comput. Sci. Educ. **13**(2), 137–172 (2003)
B Shneiderman, R Mayer, Syntactic/semantic interactions in programmer behavior: A model and experimental results. Int. J. Comput. Inf. Sci. **8**(3), 219–238 (1979). doi:10.1007/BF00977789
VJ Shute, Who is likely to acquire programming skills? J. Educ. Comput. Res. **7**(1), 1–24 (1991). doi:10.2190/VQJD-T1YD-5WVB-RYPJ
JT Townsend, FG Ashby, Methods of modeling capacity in simple processing systems, in *Cognitive theory*, ed. by J Castellan, F Restle, vol. 3 (Erlbaum, Hillsdale, N.J., 1978), pp. 200–239
V Vainio, J Sajaniemi, Factors in novice programmers' poor tracing skills. SIGCSE. Bull. **39**(3), 236–240 (2007). doi:10.1145/1269900.1268853
GL White, MP Sivitanides, Theory of the Relationships between Cognitive Requirements of Computer Programming Languages and Programmers' Cognitive Characteristics. J. Inf. Syst. Educ. **13**(1), 59–68 (2002)

BC Wilson, A Study of Factors Promoting Success in Computer Science *Including Gender Differences*. Comput. Sci. Educ. **12**(1–2), 141–164 (2002)
BC Wilson, S Shrock, *Contributing to success in an introductory computer science course: a study of twelve factors* (Paper presented at the Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education, Charlotte, North Carolina, USA, 2001)
FW Young, Multidimensional scaling. Retrieved from http://forrest.psych.unc.edu/teaching/p208a/mds/mds.html (2013)
M Yousoof, M Sapiyan, K Kamaluddin, *Reducing Cognitive Load in Learning Computer Programming*. In Proceedings of the World Academy of Science, Engineering and Technology, Vienna, 2006, pp. 259-262