Undergraduate Students' Experiences in Programming: Difficulties and Obstacles

Üniversite Öğrencilerinin Programlama Deneyimleri: Güçlükler ve Engeller

Büşra Özmen
Hacettepe University, Turkey
busraozmen@hacettepe.edu.tr

Arif Altun
Hacettepe University, Turkey
altunar@hacettepe.edu.tr

Abstract
Programming courses become prominent as one of the courses in which undergraduate students are unsuccessful especially in departments which offer computer education. Students often state that these courses are quite difficult compared to other courses. Therefore, a qualitative phenomenological approach was used to reveal the reasons of the failures of the undergraduate students in programming courses and to examine the difficulties they confronted with programming. In this scope, the laboratory practices of the Internet Programming course were observed in fall term of the 2013-2014 academic year in a university at central Anatolia. Interviews were made with 12 undergraduate students taking this course. Finally, the difficulties students experienced in the programming were determined as programming knowledge, programming skills, understanding semantics of the program, and debugging. Students emphasized that the biggest causes of failure in programming languages are lack of practice, not using algorithms and lack of knowledge. In addition, it was seen that the students who had high programming experience possess higher programming success and self-efficacy related to programming.

Keywords: P*rogramming; programming language; programming experience.*

Öz
Programlama dersleri, özellikle bilgisayar eğitimi veren bölümlerde üniversite öğrencilerinin en başarısız olduğu derslerden biri olarak öne çıkmaktadır. Öğrenciler bu dersleri diğer derslere oranla oldukça zor bulduklarını ifade etmektedirler. Bu doğrultuda, üniversite öğrencilerinin programlama derslerindeki başarısızlıklarının nedenlerini ortaya koymak ve programlama sürecinde yaşadıkları zorlukları incelemek amacıyla nitel araştırma yöntemlerinden biri olan fenomenoloji yaklaşımı kullanılmıştır. Bu kapsamda, 2013-2014 yılı güz döneminde Türkiye'de bir devlet üniversitesinde İnternet Tabanlı Programlama dersinin laboratuvar etkinlikleri gözlemlenmiş ve bu derse devam eden 12 öğrenci ile görüşme yapılmıştır. Sonuç olarak, öğrencilerin programlama sürecinde yaşadıkları zorluklar programlama bilgisi, programlama becerisi, programın mantığını kavrama ve hata ayıklama olarak belirlenmiştir. Öğrenciler, programlamadaki başarısızlıklarının en büyük nedenlerini pratik eksikliği, algoritma oluşturmama ve bilgi eksikliği olduğunu vurgulamışlardır. Ek olarak, programlama deneyimi yüksek olan öğrencilerin programlama başarılarının ve programlamaya ilişkin öz yeterlilik algılarının yüksek olduğu görülmüştür.

Anahtar Sözcükler: *Programlama; programlama dili; programlama deneyimi.*

Inroduction

In the age of information rapid changes in technology and knowledge-based economy increase the demand for people graduated from computer-related departments such as computer science, engineering and communication technology. This situation requires creation of effective learning opportunities to prepare the students who study in above-mentioned departments for the ever-changing technological environment. It was required that educators face the challenges in this process in order to train students well in terms of advanced technology and twenty-first century competencies (Law, Lee & Yu, 2010). In these popular departments, programming is one of the basic skills which are necessary to be given to the students.

Programming is one of the basic competencies that students should have in many departments such as in engineering, in computer education and instructional technologies, and in computer science. According to Lau and Yuen (2011), although programming has lost its popularity with the emergence of social networking tools, it is one of the important dimensions of the technology literacy, which is underestimated in formal schooling just as reading skill is emphasized mostly whereas writing skill is ignored in most of these institutions (Akpınar & Altun, 2014).

Programming skill has been described as an important instrument in developing higher-order thinking skills of the individual (Papert, 1991 as cited in Fessakis, Gouli & Mavrodi, 2013). It has an important role especially in developing problem solving skills (see, Ambrosio et al., 2011; Bergersen & Gustafsson, 2011) at all education levels and therefore attracting the attention of the researchers (Fessakis, Gouli & Mavrodi, 2013). In addition, programming tools are considered as powerful tools with which students can solve problems by editing, analyzing, evaluating and explaining their thoughts clearly (diSessa & Abelson, 1986). In the literature, it has been a quite established agreement that computer programming makes positive effects on students' cognitive development (Crescenzi et al., 2012; Utting et al., 2010; Clements & Sarama, 2003).

There are many factors affecting the programming success. Recently, there has been a trend related to the discovery of the predictors of programming skills and the cause of the failure in programming courses (Ferrer-Mico, Fernandez & Sanchez, 2012; Hwang et al., 2012; Shaw, 2012; Lau & Yuen, 2011; Lau & Yuen, 2009; Sivasakthi & Rajendran, 2011; Hawi, 2010; Jegede, 2009). Studies on this issue show that programming success is affected by factors such as gender (Yurdugül & Aşkar, 2013; Sullivan & Bers, 2012; Lau & Yuen, 2011), programming experience (Bergersen & Gustafsson, 2011; Lau & Yuen, 2011; Jegede, 2009), academic achievement and mathematic performance (Lau & Yuen, 2009; Ambrosio et al., 2011), self-efficacy (Jegede, 2009; Altun & Mazman, 2012) and problem solving skills (Yurdugül & Aşkar, 2013; Fessakis, Gouli & Mavroudi, 2013). Therefore, investigation of factors affecting success in programming courses can increase in student success in these courses which is mandatory and is mostly perceived as difficult.

Learning a programming language is a difficult process that requires quite a long time. Especially undergraduate programming courses are perceived as difficult by students who have basic programming knowledge because it often requires higher-order thinking skills (Tan, Ting & Ling, 2009). Studies indicate that the majority of students have difficulties in learning programming languages (Ambrosio et al., 2011; Hawi, 2010; Aşkar & Davenport, 2009). This situation has resulted in failure in programming courses (Robins, Rountree & Rountree, 2003). Students' repeated failure experience has led to a loss of excitement and interest especially towards learning programming language (Law, Lee & Yu, 2010). This situation has led researchers to study on increasing the programming success and to develop different methods in this regard. Jiau, Chen and Su (2009) have developed materials related to concretization of programming education. Rajala et al. (2008) have prepared a program visualization

tool (VILLE) by using Java. Arabacıoğlu, Bülbül and Filiz (2007) have designed an application language for teaching and turning programming logic into concrete display. Similarly, Mannila et al. (2006) have focused on the teaching of programming languages with Python.

Programming language course is still one of the most difficult courses that students fail. Although there are various demands revealed in the literature for learners to master in programming, most of the studies were conducted with computer science students in mind. In cases where computer teachers are trained for lower grades, there is a need to explore the challenges teacher trainees experience during this process. Therefore, in this study, it was aimed to reveal the reasons of the failures of the undergraduate students in programming courses at the college of education and to investigate the views about problems that they confronted with in programming.

Method

This study was designed as a phenomenological research, which aims to reveal individuals' perceptions of experiences about a phenomenon. In phenomenological research, researchers focus on a topic which they are actually aware of but about which they do not have deep knowledge (Creswell, 2007; Yıldırım & Şimşek, 2011). Hence, in this study it has been attempted to determine the reasons of failure of undergraduate students in programming courses and the difficulties they encounter.

Study group

In determining the study group, criterion-based sampling was selected from the purposeful sampling methods. According to Patton (1990), criterion-based sampling gives the possibility to obtain a wealth of information by examining the topics studied deeply. In this way, it provides great benefits to reveal lots of events and phenomenon, and their explanations. In criterion-based sampling, the study group is formed by choosing the ones that meet the determined criteria by researchers (Yıldırım & Şimşek, 2011). Study group consisted of undergraduate students taking the Internet Programming course at the department of Computer Education and Instructional Technologies (CEIT) program in a university at central Anatolia in 2013-2014 fall term. High-level expression skills and the desire to participate voluntarily were taken to into consideration while selecting the study group. Interviews were conducted with 12 students. Demographic information regarding the participants was given in Table 1.

Seven students who participated in the working group are women and five of them are men. A large proportion of these students (N=9) has graduated from vocational high schools. Achievements scores in Programming I and Programming II courses which students took in their prior terms and Internet Programming course were taken for granted in describing their programming success. Information on programming experience of the students was taken from the answers given to the question *"What was your previous programming experience?"*

Table 1
Participant Information

Student	Gender	Type of graduated high school	Programming experience	Programming success
S1	Female	Vocational	Low	High
S2	Female	Vocational	High	High
S3	Female	Vocational	Low	Low
S4	Female	General	Low	Medium
S5	Female	Vocational	Low	Medium
S6	Male	Vocational	High	High
S7	Female	Vocational	Low	Low
S8	Male	Vocational	Low	Medium
S9	Male	Technical	High	High
S10	Male	Technical	High	Low
S11	Female	Vocational	Low	Medium
S12	Male	Vocational	High	Medium

Data Collection Tools

The laboratory meetings during the Internet Programming course were observed for two weeks by one of the authors. In the first week, two observations were made for two and a half hours. Similarly the second week, two observations were made for three hours. The activities conducted in this course, the level of students' engagement in these activities and students' willingness to write a program were examined. Non-structured observation was used in this study. Observations were carried out in a laboratory. In the observation process the researcher made non-participant observation. The observer does not interfere with the process of observation in this type of observation (Glesne, 2013). Field notes from each observation were made by the researcher and were used to supplement the information collected in the interviews. All of the observations were recorded in video format. The data, collected in the observation, may be used as the data source that provides contribution and additional information to forming the study in detail (Glesne, 2013; Yıldırım & Şimşek, 2011). With the observations made in the study, it was decided on sub-objectives of the study, the study group and the place of the study. In this direction it has been decided to interview with the students regarding to their difficulties in programming.

The interviews were conducted as semi-structured and were individual interviews, based on an interview guide and comprising a series of open-ended questions. Each interview lasted approximately 15-20 min and all of them were made in the study room of the CEIT department by first author. There were eight questions in the form. The questions asked in the interview included guiding questions that students have followed in the programming, the practices they have made in the programming course and the difficulties they had confronted in this process, the effect of previous programming experience, the perception of self-efficacy related to their programming skill and wish of spending additional time for programming. The questions were designed to encourage students to give detailed information in order to ensure an effective interview. It was paid a special attention not to give a directive guidance during the interviews. All interviews were recorded on a tape recorder to be transcribed afterwards.

Data Analysis

The data attained in the study were analyzed through content analysis, which aims at bringing similar data together under certain concepts and themes. In this direction, the data are conceptualized initially, and then the concepts are arranged systematically and finally the data analyzing process is terminated with the themes formed (Yıldırım & Şimşek, 2011).

Before data analysis, all recorded data were transcribed by the researcher. The analysis was processed through qualitative data analysis software (NVivo8), which allowed coding the data obtained as a whole and taking fertile visual output at the end of the analysis. The students' statements were transcribed without any changes. Within the scope of data analysis, transcripts were initially read carefully. Then, the coding process was started and various free node lists were created. After these lists were investigated in a detail, the tree nodes were systematically grouped and categorized in a manner that is consistent within itself. At the next step, they were collected in a common tree node. In this process, it was given importance to making the most appropriate meaningful loadings to the main codes and the sub-codes relating to each main code formed. Finally, data analysis was completed once the themes had been created. Obtained codes and their frequencies were presented in tables. In addition, it is also included that the models which including the links to with a top theme of the loadings have made.

Triangulation of data collection tools increases validity by reducing bias in qualitative research (Creswell, 2008). In this study, triangulation of data is maintained by both taking students' views about the difficulties in programming process, interview data and making observations during laboratory practices. To ensure inter-rater reliability, the data was first coded and the themes created by the first author. Then, these themes and randomly selected sample statements related to themes were given two experts who have taken a course in qualitative research methods. Later, they were asked to code the documents according to the themes. A percentage agreement between two experts' reports was calculated as 81%.

Findings

The findings will be presented under the themes emerged from the analysis: difficulties encountered in programming, the reasons of failures in programming, self-efficacy for programming, and steps followed in programming. These themes will first be explained as how they showed patterns, and then, will be provided samples from the data. Table 2 shows the corresponding qualifications across the themes.

Difficulties encountered in programming

Students' views regarding the difficulties they faced in programming were gathered under four sub-themes which were named as "understanding semantics of the program", "debugging", "programming skills", and "programming knowledge". Model of current theme and its subthemes were given in Figure 1.

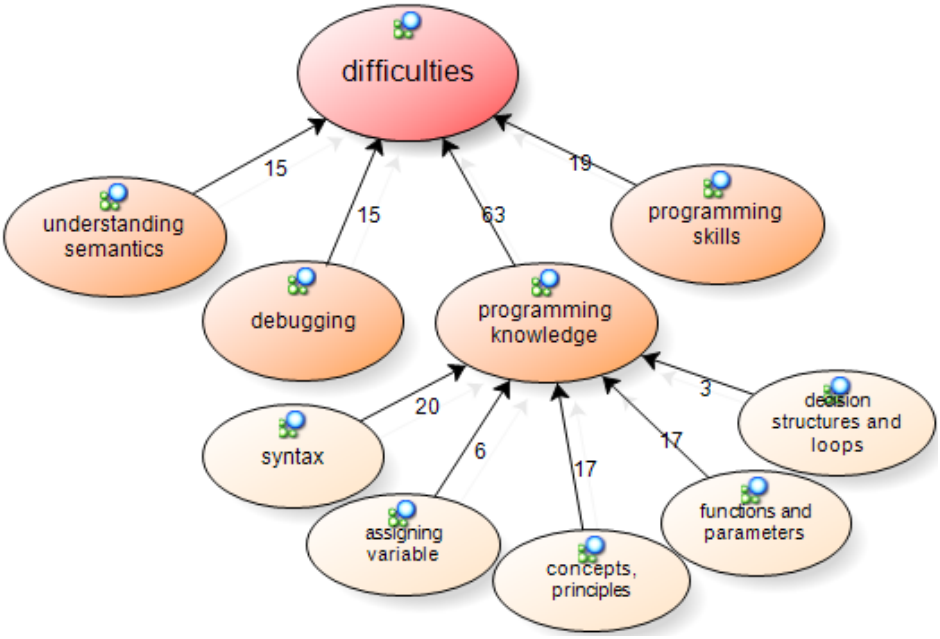

Figure 1. Difficulties Encountered in Programming

Table 2 shows the corresponding qualifications across the difficulties encountered in programming theme.

Table 2
Qualifications Across the Difficulties Encountered in Programming Theme

Difficulties encountered in programming	**Expanded descriptions**
Programming knowledge	About difficulties related to lack of knowledge
Functions and parameters	About difficulties confronted with in remembering the functions and their parameters in programming
Concepts, principles	About difficulties confronted with in knowing concepts, principles or certain facts related to programming language
Assigning variable	About difficulties confronted with in determining the variables to be used in the program and assigning them
Decision structures and loops	About difficulties confronted with in making decisions related to that which decision structures and loops will be used
Syntax	About difficulties confronted with in knowing and remembering the syntax while writing a program
Understanding semantics of the program	About difficulties confronted with understanding the semantics when writing a program code
Debugging	About difficulties confronted with in debugging errors in writing program codes which have been written earlier and he/she wrote
Programming skills	About difficulties confronted with in determining strategy to be followed while reviewing his/her programming knowledge and designing solutions to problem

It was observed that students put their emphasis the most on programming knowledge when describing the difficulties the experienced in programming. When the data within this theme were examined in a more detailed way, it was observed that "syntax" is the topic expressed mostly, and "concepts or principles", "functions and parameters", "assigning variables" and "decision structures and loops" are other topics. Frequencies of codes gotten from analysis of students' responses related to the difficulties during programming were given in Table 3.

Table 3
Codes Related to the Difficulties Confronted with and Frequencies

Codes	**f**
Programming knowledge	12
Syntax	9
Functions and parameters	7
Concepts, principles	6
Assigning variable	3
Decision structures and loops	2
Understanding the semantics of the program	8
Debugging	8
Programming skills	7

Almost all of the students experience problems in programming knowledge, and as a result of this, they have difficulty in syntax. In addition, it was found that most of the students have trouble in the topics of understanding semantics of the, debugging and programming skills. Understanding semantics of the program, on the other hand, is different from procedural knowledge. In the procedural knowledge, students can explain the necessary steps when transforming the programming knowledge. However, in this situation it is intended to describe their actual performance where they fail to accomplish this regardless of their existent procedural knowledge. Programming skills theme refers to students' ability to designing solutions to problems in programming and to determining strategy to be followed while reviewing his/her programming knowledge. Concepts or principles theme refers to declarative knowledge of concepts, principles or certain facts related to the programming language. The following statements portray this issue well:

> **S7:** *"I cannot keep codes in my mind. I grasp the logic and I say to myself that I will do it here. I can make the sequence, but I cannot write codes after memorizing them. I have to perpetually look at somewhere to write." (Programming knowledge- -concepts, principles)*
>
> **S1:** *"I read the problem. If I think that there are functions I don't know, I experience difficulties and I see it as a difficult problem" (Programming knowledge - functions and parameters)*
>
> **S4:** *"I cannot remember string functions. I cannot memorize function of counting number of lines." (Programming knowledge – functions and parameters)*
>
> **S1:** *"I haven't learnt defining array variables. I couldn't understand how to do it. Because I don't understand it alone although I searched the Internet, I have troubles in that topic now." (Programming knowledge – assigning variable)*
>
> **S3:** *"May be making loops. It is loop topic. It is boring to put an operation into a loop and to make it continuous. There are a start and an end but it is quite difficult to build something." (Programming knowledge – decision structures and loops)*

S5: *"Problems are mostly related to syntax. … It becomes either when I don't use a sign or when usage is incorrect. It becomes when I don't use the word which imply it correctly." (Programming knowledge - syntax)*

S6: *"Punctuation marks cause problems. I forget their place." (Programming knowledge - syntax)*

S4: *"Even if I understand, I can't transform algorithm to codes after forming algorithm. I can write algorithm, but then I cannot continue after that." (Programming skills)*

S5: *"I experience confusion about that I will delete it after I read one line or after I read all lines. In other words, I don't guess exactly which way I should follow." (Programming skills)*

S6: *"I don't realize immediately because error sometimes might not be at the current line. Sometimes, error might be at the first couple of lines. It makes following lines and current line wrong. I look at the current line to find the problem, but it is at other lines. It might be difficult to find it." (Debugging)*

S7: *"I can find my mistakes, but I cannot find bugs when it doesn't say 'There is an error here' at the lines where I am certain that they are correct." (Debugging)*

S4: *"It takes a quite long time to try to understand the program as a first step." (Understanding semantics of the program)*

S1: *"I don't know the solution now and I can't imagine what functions I could use". (Understanding semantics of the program)*

Reasons of students' failures in programming

Reasons of students' failures in programming were gathered under three themes which are "problems in programming process", "personal problems" and "problems in class activities". Figure 2 displays the model of current theme and its subthemes.

When the patterns within the theme of problems in the programming process were examined, the following sub-themes were observed: "not making repetition", "code editor's remark on errors", "lack of knowledge", "not writing algorithm" and "benefiting from available codes"; Similarly, "not liking programming or department", "anxiety", "prejudice" and "inattention" were the other observed themes. The problems in class activities included patterns in "studying many topics in a short time", "insufficiency of time allocated for application" and "instruction of topics not with a step-by -step method".

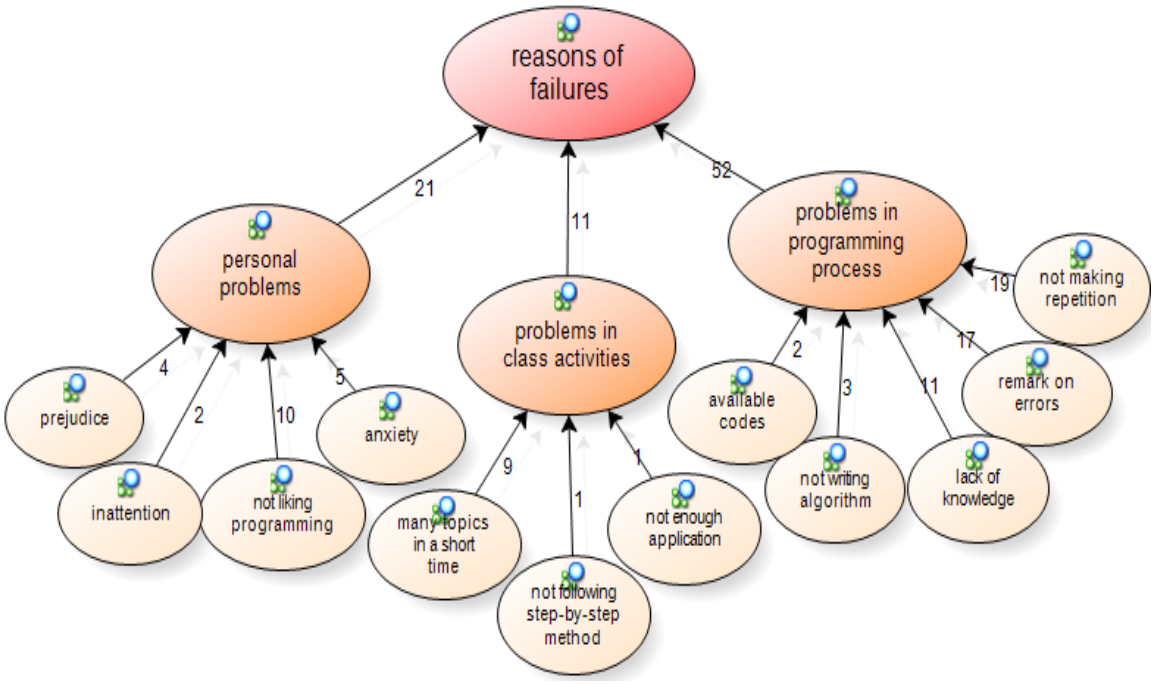

Figure 2. Reasons of Failures in Programming

Table 4 shows the corresponding qualifications across the reasons of failures in programming theme.

Table 4
Qualifications Across the Failures in Programming Theme

Reasons of failures in programming	**Expanded descriptions**
Problems in the programming process	About reasons of students' failures in programming process
Code editor's remark on errors	About code editor's remark on errors
Not making repetition	About lack of practice in programming
Lack of knowledge	About lack of knowledge about topics in programming
Not writing algorithm	About not writing an algorithm before writing the program
Benefiting from available codes	About using codes written by another person in advance
Personal problems	About reasons of failures related to individual experiences
Anxiety	About anxiety of being unsuccessful in programming
Not liking programming or the department	About not liking programming or attending (being a student at) the department in general
Prejudice against programming	About prejudicing against writing a working program
Inattention	About being inattentive while writing a program.
Problems in class activities	About reasons of failures confronted with in class activities
Insufficiency of time allocated for application	About giving lots of information on theoretical knowledge in class activities
Studying many topics in a short time	About not giving enough time on each topic in class activities
Instruction of topics without following a step-by -step method	About not giving topics step by step in class activities

Frequencies of codes related to students' failures in programming course and their frequencies were given in Table 5.

Table 5
Codes Related to Students' Failures in Programming Course and Frequencies

Codes	**f**
Problems in programming process	12
Code editor's remark on errors	11
Not making repetition	8
Lack of knowledge	5
Not writing algorithm	3
Benefiting from available codes	1
Personal problems	8
Anxiety	5
Not liking programming or department	2
Prejudice against programming	2
Inattention	2
Problems in class activities	3
Insufficiency of time allocated for application	3
Studying many topics in a short time	1
Instruction of topics without following a step-by-step method	1

It was obtained that all students attributed their failures to the problems in the programming process. Therefore, most of the students stated that code editor's remark on errors prevents them from learning codes correctly and they do not repeat topics studied in the classrooms; as a result, they experience failures. Moreover, it was found that the majority of the students emphasized personal problems. Example student views related to this issue are as follows:

> **S9:** *"In my opinion, everybody should form process in his/her mind since everybody does in this way. It is necessary to divide it to certain steps. It is taught when you start to programming. Define problem, determine needs, and etc. I do not expect any problems in programming if everybody follows these steps and if they prepare necessary algorithms before programming." (Problems in programming process-not writing algorithm)*
>
> **S1:** *"I think it is about making repetition. When person makes repetition and practice, s/he gets used to programming. You pass cognitive level, and you attain to application level. You can write without thinking, but as a first steps you should develop your skills and you should repeat." (Problems in programming process- not making repetition)*
>
> **S9:** *"It is necessary to look at your codes frequently in order to see where error is and to control each step while writing your codes. We do not have such a chance on paper. You might skip some parts on paper; or if you have syntax error and you do not assign first values to variables, errors become." (Problems in programming process - code editor's remark on errors)*

S3: *"Problems come out as a result of lack of knowledge. It is because I do not have enough knowledge." (Problems in programming process-lack of knowledge)*

S11: *"Also, there are solutions of the problems in the Internet. This is bad."* (*Problems in programming process- benefiting from available codes*)

S3: *"I do not like spending time for programming so much. I am not interested in writing a program. …if somebody tells me or if it is graded, I will do it at that time. I do not so much willingness." (Personal problems-not liking programming)*

S6*: "If I insist on that I cannot do… It becomes when I think and insist on that I cannot do that. Where do I have problem? I develop anxiety when I cannot do." (Personal problems- anxiety)*

Problems related to self-efficacy for programming

It was seen that eight students stated they could write any program even if they had to give some efforts on it. As a result, it can be said that these students have high self-efficacy. Also, it was found that these students have high success in Web-based programming course and Programming I-II course taken in prior semesters, and they have high prior experiences related to programming. Other four students stated that they could not write the program and they had to seek help for it. Example student views related to this issue are as follows:

S5: *"I can write. Normally, when someone wants me to write a program right now or when s/he says 'can you do that in this way?' I cannot do. But I do research about the requested program. I learn its usage and what I should do for it, and I can do it by trial-errors in a step-by-step method."*

S6: *"PHP or every programming language. After I understand its logic, anyone can ask any problem related to any programming language. Even if I do not know that programming language, I learn its general logic and its structure… I prepare an algorithm, concept map or flowchart. I look at the Internet for the codes of that language, and I change its codes even if I do not know. I can write. It is not a problem for me."*

S9: *"Generally, I think writing. Actually, I want to give an effort for it. Hence, I can write it in my mind. I do not know whether there was a program I could not write. I guess not."*

S7: *"I can write when some sources are on me. Otherwise, I cannot write. I can do when there are simple problems such as sum of two numbers, but I cannot do when there are quite complex problems."*

S4: *"I cannot solve the problem when you give it to me right now. I haven't still had such ability."*

Steps followed in programming

Steps the students follow in programming were gathered under five themes which are "trying to understand the problem", "preparing algorithm", "benefiting from available codes", "finding bugs" and "making repetition". Codes related to these themes and their frequencies were given in Table 6.

Table 6
Codes Related to Steps Followed while Writing Programs and Frequencies

Codes	**f**
Trying to understand the problem	8
Preparing algorithm	8
Benefiting from available codes	5
Other documents	2
Internet	2
Books	2
My own notes	1
Examples academic staff gives	1
Finding bugs	4
Making repetition	1

It was determined that the way students follow while writing programs is generally trying to understand the problem, preparing algorithm, writing and finding bugs in programming. It was obtained that eight students out of 12 put emphasis on preparing algorithm which carries significant importance in programming. Consequently, it can be said that the majority of the students try to prepare algorithm before writing program. Furthermore, 5 students stated that they benefited from the Internet, books and examples the academic staff gives. It was found that there are 4 students who try to find bugs in program after they have finished writing coding or while they are writing code. One of the students told that s/he tried to write a program by making repetitions of examples academic staff gave. Sample student views related to this issue are as follows:

> **S5:** *"Firstly, I examine documents related to the topics. How is it used, what are done, which examples are there? I make an outline of what I will do while writing codes. For example, I draw the things I will do before writing codes of a website. I say I will do it here, and this will be here. Then, I pass them to computer environment. I say 'Complete the first step, and then the second step'."*
>
> **S1:** *"Firstly, I try to understand the problem. I form an outline by taking the problem into account. I think about that which functions I can use and which ways I can follow?' After that, I start to write. While I am writing, solution appears. I write and I try to find solutions."*
>
> **S7:** *"I think about how I can solve the problem. It is necessary to have an algorithm to do it. How will I write? How will I start? Firstly, I prepare an algorithm. Then, I start to write."*
>
> **S11:** *"I form an algorithm in my mind. I draw it on the paper. While I am writing a program, I look at what is wanted, what I should do for them and how I should proceed. When this is wrong, I revise the draft."*

Conclusion

In this study, where undergraduate students' views on the causes of failure in programming courses and the problems they had encountered in programming were examined. In this regard, it has been observed that students' difficulties were mainly related to programming knowledge, programming skills, understanding semantics of the program and debugging; in addition, programming knowledge came to

the front among others. Difficulties related to programming knowledge can be listed in the following order; syntax, knowing the concepts or principles related to the programming language, remembering the functions and its parameters, defining variable and choosing the decision structures and loops that will be used in program.

The students who participated in the study stated that they had difficulty in recalling for programming codes/commands in general. This situation can be interpreted as one of the biggest obstacle in programming success resulted from lack of knowledge about program codes or producing them. Similarly, Sivasakthi and Rajendran (2011) indicated that students often had problems in code writing. Another issue which causes the students to have difficulty in programming was determined as remembering functions related to programming languages and the parameters for these functions. According to Ala-Mutka (2004), teaching the basic notions of programming after algorithms will increase programming success.

Syntax errors are one of the topics which students have difficulty while writing programs. Confusion the students experienced while using punctuation marks and while deciding their place in the code line has led to appear a considerable number of errors in the program that students wrote. This result is similar to Tan, Ting and Yang's (2011) study findings which is learning the syntax that is related to programming is one of the most important problems that the students confronted with in programming.

Unlike programming knowledge possessed by students, the strategies that they need regarding to how to use this information is defined as programming skills (Caspersen, 2007). Programming skill is one of the basic skills that students should have in programming (Holvikivi, 2010). In the present study, students, despite of having programming knowledge required, had difficulties in how they would design the program. Similarly, Tan, Ting and Yang (2011) stated that the students have problems with problem solving and separating the problems into steps in programming. Eryılmaz (2003) also stated that especially the individuals those new to programming (novice programmers) should have problem-solving skills as a prerequisite.
Students who participated in the study expressed that they have difficulties in debugging especially if written by another person. In a similar study, Tan, Ting and Yang (2011) emphasized that one of the issues that the students have difficulty is finding bugs in the program. According to Bednarik and Tukiainen (2004), debugging is directly related to previous programming experience, advanced programmers are more successful than novice programmers in debugging. Thus, it was seen in this study that the students whose programming experience is comparably higher expressed that they wouldn't find bugs in program.

Another issue emphasized in the study is the reasons of failures of the students in programming courses. Student' experiences related to this issue were gathered under three themes. These themes have been ranked in sequence in descending order: problems *in the programming process*, personal problems, and problems they faced during in-class activities. The scope of the problems related to in the process of programming topics that lead to failure are listed as not making repetition, code editor's remark on errors, lack of knowledge, not writing algorithm and benefiting from available codes. Within the scope of the personal problems are dislike programming or attending the department, prejudice against programming, anxiety and inattention. Finally, the scope of the problems they faced during in-class activities are listed as studying many topics in a short time, insufficiency of time allocated for application and instruction of topics without following a step-by-step method.

The majority of students stated that the biggest reasons of their failure in programming were related to themselves since they do not work hard enough in programming. In addition, not allocating extra time to programming and to course activities and assignments, as well as insufficient repetitions about

the programs learned in the lessons were articulated among the others. In the literature, it has been emphasized that undergraduate students need lots of practice to increase their programming skills (Law, Lee & Yu, 2010). Similarly, Hawi (2010) determined that the most important factors that cause students to fail are lack of study, lack of effort and lack of practice. Accordingly, studying on a computer programming language requires making many activities such as reading textbooks and reference books recommended, using online libraries to writing code, learning syntax and logical concepts, analyzing available programs and modifying them.

Students whose previous programming experience are higher (advanced programmers) highlighted that programming performance is dependent on preparing an algorithm. Similarly, Bednarik and Tukiainen (2004) stated that while advanced programmers build hypotheses before writing a program, novice programmers start with writing the program directly. Preparing algorithm become prominent as the primarily issue for especially novice programmers who begin learning a new programming language (Eryılmaz, 2003). Because in the process of preparing an algorithm, flow charts which determine the steps to be followed and strategies to be used in the solution of the problem are created. According to Ala-Mutka (2004), algorithm training which is given in the programming process helps teaching the programming language more easily and in a simple manner. However, it has been seen that developing the algorithmic thinking is one of the biggest problems in programming courses (Ziatdinov & Musa, 2012), because preparing algorithms is perceived as a difficult process and time-consuming by the students.

Other important reasons of the failure that the students confronted with in programming are prejudice against programming and anxiety. In this study it was observed that while the students who like programming reported that they often spend extra time to their self-development, find alternative ways to solve the problem or write more qualified programs; on the other hand, the students who have high level of prejudice against programming feel angry and stressed out when writing programs. Thus, they don't want to spend extra time on programming. Students who have high level of programming anxiety don't want to learn programming, because they think that programming is a difficult and boring process (Tan, Ting & Ling, 2009).

According to another result obtained in the study, it was seen that advanced programmers have a higher level of programming performance. In the literature, there are studies related to that advanced programmers are more successful than novice programmers in programming (Lau & Yuen, 2011; Liao & Bright, 1991). This situation can be interpreted as that lack of prior knowledge and experience is another reason of the reason of the students' failure. On the other hand, in this study, it was realized that the students who have graduated from vocational school failed although they were took many programming lessons. This situation supports the findings of that programming experience has a nondirective effect on programming success (Bergersen & Gustafsson, 2011; Jegede, 2009). According to Bergersen and Gustafsson (2011), programming experience affects primarily programming knowledge, and then programming knowledge affects programming success. Self-efficacy is considered as the other mediator variable that affects the success of programming in conjunction with experience. According to this, students' preliminary experiences in programming increase their self-efficacy, and perceived self-efficacy increases programming success (Jegede, 2009). In the literature, it was indicated that there is a positive correlation between students' self-efficacy and the number of programming courses taken by them (Altun & Mazman, 2012; Jegede, 2009), and the number of years in programming affects their self-efficacy significantly (Altun & Mazman, 2012; Aşkar & Davenport, 2009).In this study, it was observed that students who had considerably higher level of programming success also had a higher level of self-efficacy as well. These students defined programming as that it is actually an easy process as long as necessary repetitions are made and it is started with algorithm before writing the program. Therefore, the students believe that they can write program codes successfully if they take

enough time. This finding is similar to the results of the studies which found that perceived self-efficacy increases the programming success (Cegielski & Hall, 2006).

As a result, to get all the benefits of programming, effective programming education should be given especially in higher education institutions (Fessakis, Gouli & Mavroudi, 2013). It is highlighted that the current curricula should be reconstructed by increasing lesson hours of programming courses and widening their scope. Moreover, more emphasis is called on teaching programming (Akpınar & Altun, 2014); hence, programming education should be carried out taking into account aforementioned problems and present situation.

Further research could determine the reasons of the students' failures in visual programming courses. Also, introspective methods could be used with the aim of better documenting the obstacles and difficulties that the students confronted with in programming. Different programming languages have different application features, code brevity, and extensibility and so on. Therefore, the usage of alternative programming languages would be valuable for demonstration the differences between programming languages and comparison of the results.

References

Akpınar, Y., & Altun, A. (2014). Bilgi toplumu okullarında programlama eğitimi gereksinimi. *İlköğretim Online (13)*1, 1-4.

Ala-Mutka, K. (2004). Problems in learning and teaching programming. *Institute of Software Systems, Tampere University of Technology.*

Altun, A., & Mazman, S. G. (2012). *Programlamaya İlişkin Öz Yeterlilik Algısı Ölçeğinin Türkçe Formumun Geçerlilik ve Güvenirlik Çalışması.* Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi, Kış 2012, 3(2), 297- 308.

Ambrosio, A. P., Costa, F. M., Almeida, L., Franco, A., & Macedo, J. (2011). Identifying cognitive abilities to improve CS1 outcome. *Frontiers in Education Conference (FIE).* , 12-15 October.

Arabacıoğlu, T., Bülbül, H. İ., & Filiz, A. (2007). Bilgisayar programlama öğretiminde yeni bir yaklaşım. *Akademik Bilişim 2007,* Kütahya Dumlupınar Üniversitesi.

Aşkar, 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).

Bednarik, R., & Tukiainen, M. (2004). Visual attention and representation switching in java program debugging: a study using eye movement tracking. In Proceedings of 16th Annual Psychology of Programming Interest Group Workshop (PPIG'04), Institute of Technology Carlow, Ireland, April 5-7, 2004, pp. 159-169.

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.

Caspersen, M. E. (2007). *Educating Novices in the Skills of Programming.* (PhD), University of Aarhus Denmark.

Cegielski, C. G., & Hall, D. J. (2006). What makes a good programmer? *Communications of the ACM, 49*(10), 73-75.

Clements, D., & Sarama, J. (2003). Strip mining for gold: research and policy in educational technology – a response to "fool's gold". AACE Journal. ISSN: 1551-3696, 11(1), 7–69, Association for the Advancement of Computing in Education, Norfolk, VA, USA.

Crescenzi, P., Malizia, A., Verri, M. C., Diaz, P., & Aedo, I. (2012). Integrating algorithm visualization video into a first-year algorithm and data structure course. *Educational Technology and Society*, *15*(2), 115-124.

Creswell, J. W. (2008). *Educational research: Planning, conducting, and evaluating quantitative and qualitative research (3rd ed.).* Upper Saddle River, New Jersey: Pearson Education, Inc.

Creswell, J. W. (2007). *Qualitative Inquiry and Research Design: Choosing among five approaches* (2nd ed.). Thousand Oaks, CA: Sage.

diSessa, A.A., & Abelson, H.(1986). Boxer: A reconstructible computational medium. *Communications of the ACM, 29*(9), 859–868.

Eryılmaz, S. (2003). *Algoritma tasarlama ve programlamaya giriş*. Ankara: Detay Yayıncılık.

Ferrer-Mico, T., Prats-Fernandez, M. A., & Redo-Sanchez, A. (2012). Impact of Scratch programming on students´ understanding of their own learning process. *Procedia - Social and Behavioral Sciences 46* (2012), 1219-1223.

Fessakis, G., Gouli, E., & Mavrodi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers and Education 63 (2013),* 87-97.

Glesne, C. (2013). *Nitel Araştırmaya Giriş* (Çeviri Editörleri: Ali Ersoy & Pelin Yalçınoğlu). 2. Baskı. Ankara: Anı Yayıncılık.

Hawi, N. (2010). Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers and Education 54* (2010), 1127-1136.

Holvikivi, J. (2010). Conditions for Successful Learning of Programming Skills. In N. Reynolds & M. Turcsányi-Szabó (Eds.), *Key Competencies in the Knowledge Society.* 324, 155-164. Springer Berlin Heidelberg.

Hwang W.Y., Shadiev, R., Wang C. Y., & Huang, Z. H. (2012). A pilot study of cooperative programming learning behavior and its relationship with students' learning performance. *Computers and Education 58* (2012), 1267–1281.

Jegede, P. O. (2009). Predictors of java programming self–efficacy among engineering students in a Nigerian University. *International Journal of Computer Science and Information Security (IJCSIS), 4*(2).

Jiau, H. C., Chen, J. C., & Su, K. F. (2009). Enhancing self-motivation in learning programming using game-based simulation and metrics. *IEEE Transactions on Education, 52*(4), 555-562.

Lau, W. W. F., & Yuen, A. H. K. (2009). Exploring the effects of gender and learning styles on computer programming performance: implications for programming pedagogy. *British Journal of Educational Technology*, 40(4), 696-712.

Lau, W. W. F., & Yuen, A. H. K. (2011). Modeling programming performance: Beyond the influence of learner characteristics. *Computers and Education, 57*(1), 1202-1213.

Law, K., Lee, V., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers and Education 55* (2010), 218-228.

Liao, Y. C., & Bright, G. W. (1991). Effects of computer programming on cognitive outcomes: a meta-analysis. Journal of Educational Computing Research, 7(3), 251–266.

Mannila, L., Peltomaki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. Computer Science Education, 16(3), s:211-227.

Papert, S. (1991). Mindstorms: Children, computers and powerful ideas. Athens: Odysseas Publications (in Greek).

Patton, M. Q. (1990). *Qualitative evaluation and research methods* (2nd ed.). Newbury Park, CA: Sage Publications.

Rajala, T., Laakso, M.J., Kaila, E., & Salakoski, T. (2008). Effectiveness of program visualization: a case study with the VILLE tool. *Journal of Information Technology Education: Innovations in Practice, 2008*(7), 15-32.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172.

Shaw, R. S. (2012). A study of the relationships among learning styles, participation types, and performance in programming language learning supported by online forums*. Computers and Education 58* (2012), 111–120.

Sivasakthi, M., & Rajendran, R. (2011). Learning difficulties of 'object-oriented programming paradigm using Java': students' perspective. Indian Journal of Science and Technology, 8(4), 983-985.

Sullivan, A., & Bers, M. U. (2012). Gender differences in kindergarteners' robotics and programming achievement*. International Journal of Technology and Design Education, 23*(3), 691-702.

Tan, P. H., Ting, C. Y., & Ling, S. W. (2009). Learning difficulties in programming courses: Undergraduates' perspective and perception. *2009 International Conference on Computer Technology and Development,* Kota Kinabalu, Malaysia.

Utting, I., Cooper, S., Kölling, M., Maloney, J., & Resnick, M. (2010). Alice, Greenfoot, and scratch - a discussion. *ACM Transactions on Computing Education*, 10(4), 1-11.

Yıldırım, A., & Şimşek, H. (2008). *Sosyal Bilimlerde Nitel Araştırma Yöntemleri* (6. Baskı), Ankara: Seçkin Yayınevi.

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

Ziatdinov, R., & Musa, S. (2012). Rapid mental computation system as a tool for algorithmic thinking of elementary school students development. *European Researcher*, 25(7), 1105-1110.

GENİŞLETİLMİŞ ÖZ

Programlama dili öğrenme oldukça uzun bir zaman gerektiren zor bir süreçtir. Özellikle lisans düzeyindeki programlama dersleri çoğunlukla üst düzey düşünme becerileri gerektirdiği için genellikle giriş seviyesinde programlama bilgisine sahip öğrenciler tarafından oldukça zor olarak algılanmaktadır (Tan, Ting ve Ling, 2009; Gültekin, 2006). Yapılan çalışmalar öğrencilerin çoğunun programlama dili öğrenmede güçlük çektiğini göstermektedir (Başer, 2012; Ambrosio vd., 2011; Hawi, 2010; Aşkar ve Davenport, 2009). Bu durum öğrencilerin programlama derslerinde başarısız olmasıyla sonuçlanmaktadır (Robins, Rountree ve Rountree, 2003; Baldwin ve Kuljis, 2000). Öğrencilerin tekrarlayan başarısızlık

deneyimi, özellikle programlama dili öğrenmeye karşı heyecan ve ilgilerini kaybetmelerine neden olmaktadır (Law, Lee ve Yu, 2010). Bu durum araştırmacıların programlama başarısını artırmak üzere çalışmalar yapmaları ve bu konuda farklı yöntemler geliştirmelerine neden olmuştur. Ancak yapılan bu çalışmalar yeterli olmamıştır. Programlama dilleri dersi günümüzde hala öğrencilerin en çok zorlandığı ve başarısız olduğu derslerden biri olmaya devam etmektedir. Buradan hareketle, yapılan çalışma ile üniversite öğrencilerinin programlama derslerindeki başarısızlıklarının nedenlerinin ortaya koyulması ve programlama sürecinde yaşadıkları problemlere ilişkin görüşlerinin incelenmesi amaçlanmıştır.

Çalışma nitel araştırma yöntemlerinden fenomenoloji yaklaşımı ile desenlenmiştir. Çalışma grubu, 2013-2014 güz yarıyılında Türkiye'de bir devlet üniversitesinde İnternet Tabanlı Programlama dersine devam eden öğrenciler arasından seçilmiştir. Çalışma grubunun belirlenmesinde amaçlı örnekleme yöntemi çeşitlerinden ölçüt örnekleme kullanılmıştır. Ölçüt olarak, öncelikle ifade yeteneği yüksek 18 öğrenci seçilmiş, daha sonra çalışmaya katılmaya gönüllü olduğunu ifade eden 12 öğrenci ile görüşme yapılmıştır. Verilerin toplanma sürecinde öncelikle dersin laboratuvar etkinlikleri iki hafta süreyle araştırmacı tarafından yapılandırılmamış gözlem tekniği ile gözlemlenmiştir. Katılımcı olunmayan gözlem anlayışına uygun biçimde yapılan gözlemlerin tümü video kaydına alınmıştır. Yapılan gözlem ile yürütülmesi planlanan çalışmanın alt amaçları, çalışma grubu, çalışmanın yapılacağı ortam hakkında karar verilmiştir. Bu doğrultuda öğrencilerle, programlama sürecinde yaşadıkları zorluklara ilişkin görüşme yapılması kararlaştırılmıştır. Görüşmeler yarı yapılandırılmış görüşme türüne uygun olarak hazırlanmış ve bireysel görüşme şeklinde gerçekleştirilmiştir. Tümü sesli kayıt altına alınmıştır. Elde edilen veriler içerik analizine uygun olarak çözümlenmiştir. Kodlamalara ilişkin değerlendiriciler arası güvenirliği sağlamak amacıyla, öncelikle veriler ilk yazar tarafından kodlanmış ve temalar oluşturulmuştur. Daha sonra, bu temalar ve temalara ilişkin rastgele seçilen örnek cümleler nitel araştırma yöntemleri dersi almış iki uzmana verilmiş ve kodlarla örnek cümleleri eşleştirmeleri istenmiştir. Uyum yüzdesi %81 olarak hesaplanmıştır.

Öğrencilerin programlama sürecinde yaşadıkları zorlukların programlama bilgisi, programlama becerisi, programın mantığını kavrama ve hata ayıklama üzerinde yoğunlaştığı görülmüştür. Araştırmaya katılan öğrenciler, büyük çoğunlukla programlamaya ilişkin genel kavram ve ilkeleri hatırlamakta zorlandıklarını ifade etmişlerdir. Öğrenciler, gerekli programlama bilgisine sahip olmalarına rağmen programı nasıl tasarlayacaklarına ilişkin zorluk yaşadıklarını da belirtmektedirler. Bednarik ve Tukiainen'e (2004) göre, bir hata ayıklama programlama deneyimiyle doğrudan ilişkilidir, programlama deneyimi yüksek olan öğrenciler düşük olanlara göre hata ayıklamada daha başarılıdırlar. Nitekim mevcut çalışmada da önceki programlama deneyimi yüksek olan öğrencilerin programdaki hataları bulmakta zorlanmayacakları yönünde görüş belirttikleri görülmüştür. Öğrencilerin büyük çoğunluğu, programlama konusunda yeterince çalışmamalarını, ders kapsamında yapılan etkinlikler ve verilen ödevler dışında programlama için ek zaman ayırmamalarını ve verilen programları tekrar etmemelerini programlama dillerindeki başarısızlıklarının en büyük nedeni olarak belirtmişlerdir. Ayrıca, programlama deneyimi yüksek olan öğrenciler, programlama başarısının algoritma oluşturmaya bağlı olduğunu vurgulamışlardır. Algoritma oluşturma, özellikle bir programlama dilini yeni öğrenmeye başlayan öğrencilerin en başta öğrenmesi gereken konu olarak öne çıkmaktadır (İmal ve Eser, 2009; Eryılmaz, 2003). Ancak öğrenciler, algoritma oluşturmayı vakit alıcı ve zor bir süreç olarak algıladıkları için (Futschek ve Moschitz, 2010) programlama derslerinde algoritmik düşünmeyi geliştirme en büyük problemlerden biri olarak görülmektedir (Ziatdinov ve Musa, 2012). Ek olarak, programlamayı sevdiğini belirten öğrencilerin genellikle kendilerini geliştirmek, problemin çözümünde alternatif yollar bulmak veya daha nitelikli programlar yazmak için ekstra zaman harcadıkları görülürken; programlamaya ilişkin ön yargısı olan öğrencilerin program yazarken kendilerini sinirli ve stresli hissettikleri ve programlama için ek vakit ayırmak istemedikleri görülmüştür. Bir diğer sonuç, programlama deneyimi yüksek olan öğrencilerin programlama başarısının yüksek olmasıdır. Ancak, bazı öğrencilerin meslek lisesi mezunu olmaları nedeniyle birçok programlama dersi almalarına rağmen başarısız oldukları görülmüştür. Bergersen ve Gustafsson'a (2011) göre

öğrencilerin programlama deneyimi öncelikli olarak programlama bilgisine, programlama bilgisi de programlama başarısına etki etmektedir. Deneyim ile birlikte programlama başarısını etkileyen bir diğer ara değişken ise öz yeterlilik olarak kabul edilmektedir. Buna göre öğrencilerin programlamaya ilişkin sahip oldukları ön deneyimler öz yeterliliklerini artırmakta, öz yeterlilik algısı ise programlama başarısını artırmaktadır (Jegede, 2009). Çalışmada programlamaya ilişkin öz yeterlilik algısı yüksek olan öğrencilerin sayısının diğerlerine göre daha fazla olduğu ve programlama başarısı yüksek öğrencilerin aynı zamanda yüksek düzeyde öz yeterliliğe sahip oldukları belirlenmiştir.

Sonuç olarak, öğrencilerin programlama sürecinde yaşadıkları güçlükleri ve karşılaştıkları engelleri en aza indirebilmek için özellikle yükseköğretim kurumlarında uygun programlama eğitimi verilmesi gerekmektedir (Fessakis, Gouli ve Mavroudi, 2013). Mevcut müfredatın programlama derslerinin saatlerinin artırılması ve bu derslerin kapsamının genişletilmesi suretiyle yeniden düzenlenmesi gerektiği vurgulanmaktadır. Ayrıca, programlama eğitimine daha fazla önem verilmelidir (Akpınar & Altun, 2014); bu kapsamda, programlama eğitimi söz konusu sorunlar ve mevcut durum dikkate alınarak yapılmalıdır.