

**DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES**

**DEVELOPING A COMPUTER-AIDED
INSTRUCTION APPLICATION TARGETING
UNIVERSITY STUDENTS**

by
Meltem YILDIRIM

**April, 2013
İZMİR**

**DEVELOPING A COMPUTER-AIDED
INSTRUCTION APPLICATION TARGETING
UNIVERSITY STUDENTS**


**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor Philosophy
in Computer Engineering, Computer Engineering Program**

**by
Meltem YILDIRIM**


**April, 2013
İZMİR**

Ph.D. THESIS EXAMINATION RESULT FORM

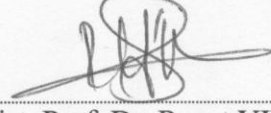
We have read the thesis entitled “**DEVELOPING A COMPUTER-AIDED INSTRUCTION APPLICATION TARGETING UNIVERSITY STUDENTS**” completed by **MELTEM YILDIRIM** under supervision of **PROFESSOR DR. ALP KUT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.


Prof. Dr. Alp KUT

Supervisor


Prof. Dr. Yalçın ÇEBİ

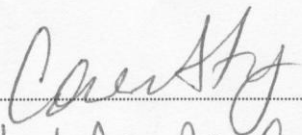
Thesis Committee Member


Assist. Prof. Dr. Reyat YILMAZ

Thesis Committee Member


Prof. Dr. Ahmet KASELİ

Examining Committee Member


Yrd. Doç. Dr. Coşkun ERER AYAY

Examining Committee Member


Prof. Dr. Ayşe OKUR

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to express my gratitude my advisor, Prof. Dr. Alp KUT for his guidance, support and friendship lead to the successful completion of my thesis.

I extend my thanks to the members of my committee, Prof. Dr. Yalçın ÇEBİ, and Asst. Prof. Dr. Reyat YILMAZ for their useful comments and suggestions during my study.

In addition, I would like to acknowledge the equipment support from the Dokuz Eylul University BAP with 2011.KB.FEN.34 project number for my doctoral study.

Finally, I would like to thank to my all family; to my parents, Ayten and Mehmet YILDIRIM for their support to date, especially to my partner, Aytek EKİCİ for his contributions and his encouragement during my study and most special thank to my little friend, Deniz EKİCİ for pretty gaps during the writing thesis and her luck.

Meltem YILDIRIM

DEVELOPING A COMPUTER-AIDED INSTRUCTION APPLICATION TARGETING UNIVERSITY STUDENTS

ABSTRACT

Over the years, many various studies realized in education to support learning-teaching process. Learning algorithm and programming depending on mental concern is often complex and difficult to understand for students. In general, to overcome those difficulties and help students had better understand the subject several educational tools and methods have been developed.

In this study, the introduced model assists students and instructors. Students can practice and assist themselves to learn algorithms and programming concepts. Instructors can use the tool during their teaching classes and get inspired by the data gathered. An educational tool named Algolyzer has been developed for this learner-centered model. Algolyzer depends on finding a solution to implement an algorithm for a predefined algorithmic problem. Students can create algorithmic steps using visual interface that students do not face with programming language syntax issues but only focus on the possible solutions. In addition to this, Algolyzer is also a helper utility for the instructors with giving information about the miscomprehension parts in the teaching process. Instructors can have detailed information on where students need more help, what are the lacking parts using the detailed logs of student activities.

The students of Dokuz Eylül University Computer Engineering and Computer Programming Department have used Algolyzer. Usage data has been examined and evaluated at the end of the study and obtained results have been shared with the instructors. The surveys that targeted the users of Algolyzer and model and the feedbacks prove that support the learning process effectively.

Keywords: Programming learning, educational tool, simplify algorithm learning, error detection, code generation.

ÜNİVERSİTE ÖĞRENCİLERİNİ HEDEF ALAN BİLGİSAYAR DESTEKLİ BİR ÖĞRETİM UYGULAMASI GELİŞTİRME

ÖZ

Öğrenme öğretme sürecini desteklemek üzere yıllar içinde birçok farklı çalışma gerçekleştirilmiştir. Öğrenciler için zihinsel işlerle ilişkili olan algoritma ve programlamanın öğrenilmesi çoğu kez karmaşık ve zor olmuştur. Genelde bu zorlukları aşmak ve öğrencilerin daha iyi anlayabilmesini sağlamak için farklı metotlar ve eğitim araçları geliştirilmiştir.

Bu çalışmada, öğrencilere algoritma ve programlamanın öğrenilmesine yardımcı ve öğretmenlerin de kendi öğretim süreçlerinde yardımcı olacak ve toplanacak veri ile esin kaynağı oluşturabilecek bir model tanıtılmaktadır. Bu öğrenci merkezli model için Algolyzer adı verilen bir eğitim aracı geliştirilmiştir. Geliştirilen yazılım aracı önceden tanımlanmış bir algoritmik problemin algoritmasını gerçekleştirerek çözümünün bulunmasına dayanmaktadır. Öğrenciler programlama dilinin sözdiziminden kaynaklanacak hata ve sorunlar ile karşılaşmadan görsel arayüzü kullanarak algoritmik basamakları oluşturabilirler. Aynı zamanda Algolyzer eğitim süreci içerisinde kavranamayan bölümlerle ilgili bilgi verdiği için, öğretmenler için yardımcı bir bileşendir. Sistemin öğrencilerin aktivitelerini kaydetmesi ile öğretmenler öğrencilerin daha fazla yardıma ihtiyaç duydukları ya da eksik kalan bölümleri hakkında detaylı bilgiye sahip olmaktadır.

Dokuz Eylül Üniversitesi Bilgisayar Mühendisliği ve Bilgisayar Programcılığı Bölümlerinde bu uygulama kullanılmıştır. Çalışmanın sonunda öğrencilerin kullanım bilgileri değerlendirilmiş ve elde edilen sonuçlar öğretmenlerle paylaşılmıştır. Model ve Algolyzer kullanıcılarına yönelik yapılan anketler ve alınan geri bildirimler öğrenme sürecine katkı sağladığını göstermektedir.

Anahtar sözcükler : Programlamayı öğrenme, eğitim aracı, algoritma öğrenmeyi kolaylaştırma, hata bulma, kod üretimi.

CONTENTS

Page

PhD. THESIS EXAMINATION RESULT FORM	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZ	v
LIST OF FIGURES	ix
LIST OF TABLES	x
CHAPTER ONE - INTRODUCTION	1
1.1 Problem Definition	2
1.2 Contributions of Thesis	3
1.3 Aims and the Scope	4
1.4 Thesis Organization	5
CHAPTER TWO - LEARNING MODEL & RELATED WORKS	7
2.1 Constructivism	7
2.2 Related Works	8
2.2.1 Introducing Algorithm and Programming with Didactic Scenarios	9
2.2.2 ELP- Environment for Learning to Program	10
2.2.3 Web-Based Environment Depending on Activities	10
2.2.4 Programming Teaching Tool with Filling Fields	11
2.2.5 Simplifying Algorithm Learning Using Serious Games	12
CHAPTER THREE - EDUCATIONAL TOOLS.....	13
3.1 Educational Tool for Program Tracing:ProTracer	14
3.2 Educational Software: CGRAPHIC	15
3.3 Educational Tool for Understanding Algorithm Building	16
3.4 Tools for OOP	17
3.4.1 TeachingOOP: ObjectKarel	17
3.4.2 TeachingOOP: AEIOU	18
3.5 Studies for Specific Algorithms.....	18
3.5.1 Learning of Sorting Algorithms	19

3.5.2 Learning Tool of Genetic Algorithm	19
3.6 Evaluation and Comparison of Tools.....	19
CHAPTER FOUR - DESIGN AND IMPLEMENTATION OF NEW	
INTRODUCED EDUCATIONAL TOOL: ALGOLYZER	22
4.1 General System Design.....	22
4.1.1 Operational Level of System.....	24
4.1.2 Infrastructural Level of System	24
4.2 Library Module of Algolyzer	25
4.2.1 Processor.....	25
4.2.2 Services	26
4.2.3 Statements.....	27
4.3 Data Module	28
4.4 Code Generation	29
4.5 Statement Details	30
4.6 User Interface	31
CHAPTER FIVE - CAPABILITIES OF ALGOLYZER.....	32
5.1 Developed Tool Position in Learning Process.....	33
5.2 Base Components of Interface	34
5.2.1 Problem Part of the Developed Tool.....	35
5.2.2 Operation Part of the Developed Tool.....	37
5.2.3 Code Part of the Developed Tool	38
5.3 Multi Programming Language Support	38
5.4 Language Support.....	38
5.5 Error Detection.....	39
5.6 The First Version of the Developed Tool.....	41
CHAPTER SIX - RESULT AND EVALUATION OF ALGOLYZER USAGE.	43
6.1 Engage Students to Use Algolyzer	43
6.2 General Statistics	44
6.3 Distributions of Students Errors During Programming	46
6.3.1 Error Distributions on Base Topics.....	46
6.3.2 Error Distributions on Subfield Topics.....	47
6.4 Individual Evaluation & Distribution of Multiple Errors.....	50

6.4.1 Apriori Algorithm Result	51
6.5 Students Evaluation of Algolyzer	54
6.6 General Evaluation	58
CHAPTER SEVEN - CONCLUSIONS	60
REFERENCES	63
APPENDICES.....	70
A.List of Abbreviations	70
B.Class Diagram Of Algolyzer.....	71
C.Algolyzer Usage	73
D.Previous Version of The Study	76

LIST OF FIGURES

	Page
Figure 2.1 General view of application.....	11
Figure 3.1 Program tracing interface of ProTracer.....	15
Figure 4.1 Main operations of general system.....	23
Figure 4.2 General system views on operational level	24
Figure 4.2 General system frameworks	25
Figure 4.4 Class diagram of processor.....	25
Figure 4.5 Services of Algolyzer.....	27
Figure 4.6 Statement classes.....	28
Figure 4.7 Data class diagram.....	29
Figure 4.8 Generate code of processor.....	30
Figure 4.9 Statements and subfields.....	30
Figure 4.10 General view of developed tool.....	31
Figure 5.1 Ratios of difficulties in programming.....	32
Figure 5.2 Developed tool position in learning process.....	34
Figure 5.3 Workflow of the base components.....	34
Figure 5.4 General view of GUI.....	35
Figure 5.5 Error detection mechanism.....	39
Figure 5.6 Sample screen of error detection.....	40
Figure 5.7 An overview of previous version of application.....	42
Figure 6.1 Error rates of base operations for Study A.....	46
Figure 6.2 Error rates of base operations for Study B.....	47
Figure 6.3 Error distribution based on the subfields for Study A.....	49
Figure 6.4 Error distribution based on the subfields for Study B.....	50
Figure 6.5 Multiple error rates for Study A.....	53
Figure 6.6 Questionnaire of Algolyzer usage.....	54
Figure 6.7 Students feedback on doing more practice.....	56
Figure 6.8 Students feedback on effectiveness of error messages.....	56
Figure 6.9 Students feedback on support degree of dividing operations.....	57
Figure 6.10 Students feedback on support degree of independent of PL.....	57
Figure 6.11 Students feedback on important feature of Algolyzer.....	58

LIST OF TABLES

	Page
Table 3.1 Features of previous educational tools in literature.	20
Table 5.1 Problem repository of the system.....	36
Table 5.2 Operation table.....	37
Table 5.3 Types of errors made by students during the programming..	39
Table 6.1 Experimental groups of study assessment.	44
Table 6.2 General Statistics of System Usage for Study A.....	45
Table 6.3 General Statistics of System Usage for Study B.....	45
Table 6.4 Subfield error rates of Study A..	48
Table 6.5 Subfield error rates of Study B.	49
Table 6.6 Relations of multi errors and rates for Study A..	51
Table 6.7 Relations of multi errors and rates for Study B..	54

CHAPTER ONE

INTRODUCTION

Over the years many various studies realized to support learning-teaching process in imparting education. Studies on teaching algorithms, programming, and data structures related to mental task are one of those works. Algorithm and programming is the main course in computer science and computer engineering education. Learning algorithm and programming is often complex and difficult to understand for students and they often face difficulties on the main courses. Teaching and learning programming has never been an obvious process because of that programming is a skill and difficult for student due to needs on deeply comprehension. Therefore, studies on learning programming skills are important. There are some studies on motivation of learning programming skills (Jerez, Bueno, Molina, Urda, & Franco, 2012).

Simplifying an algorithm learning process is quite significant for both students and instructors. To overcome the learning problem and help students better learn and understand algorithms, instructors are using different methods during education process. Educational tools are playing critical roles in education process (Jain, Singhal, & Gupta 2010). Visualizations can be used in teaching to support learners for understanding the abstract and structure (Taherkhani, Korhonen, & Malmi, 2010). In imparting education, educational tools such as visualizations, charts, simulations, online tools and any other proved or experimental tool can be used to improve and create more effective teaching and learning sessions (Shabanah, & Chen, 2009; Sutinen, Tarhio, & Terasvirta, 2003; Kordaki, 2010). Even there are some studies on exploring these features effectiveness (Hundhausen, Douglas, & Stasko, 2002) with some questions (Hundhausen, & Brown, 2008). In addition, there are also some studies on difficulties of learning and teaching programming (Lahtinen, AlaMutka, & Järvinen, 2005; Milne, I., & Rowe G., 2002).

In this thesis, the study has two phases, which will be explained in details. The first phase is new introduced educational tool-Algolyzer, which will help students to

better learn and understand algorithms and programming. The second phase of the study is an experimental usage of application by students, and evaluation of the records including user actions and errors data collected during the students' usage.

1.1 Problem Definition

Learning programming has never been an obvious process associated with programming is a skill requires practice. Students need to realize more programming practice to enhance their programming knowledge (Ng, Choy, Kwan, & Chan, 2005).

Algorithm course tries to teach algorithmic thinking and basics of programming and affects the students directly during their education cycle. While trying to teach basics of programming, instructor must also be sure that students are motivated (Holvikivi, 2010). Selecting the methodology that teaches using a programming language often fails consequence of learning the syntax of the selected language, which is not the primary goal in the process. In this case, students will dive into programming language syntax rather than focusing on algorithmic thinking. Specially improving algorithmic thinking and passing to abstract reasoning is challenging process and this requires huge effort. With starting from the importance of this topic, various studies with Bachelor Computer Science and similar department' students were realized on thinking like computer scientist's skills and abstraction level of students (Perrenet, Groote, & Kaasenbrood, 2005).

In this study, an inference done depending on the experience of algorithms and programming courses and an appropriate model produced with considering the referred points of the previous works. We approved that many students meet difficulties on learning algorithms and programming level and solving problems in an algorithmic way. Students have to compete with programming language syntax and development environments while they are working on improving the algorithmic logic. This conducts students away from the learning. Getting into the hang of writing programming code is complicated for students; if programming practices are insufficient. Another base point on this area is that focusing on understanding and

solving single problem is too important (Müldner, Shakshuki, & Kerren, 2008). Depending on the observations, this study should consider the difficulties in algorithms and programming courses and being a mental task for students. This study propose a new method with identifying the most important and ineffective parts of the existing process, which provide a supporting user-friendly interface to generate programming code independent from programming language syntax and focusing only the algorithmic thinking.

1.2 Contributions of Thesis

This study provides a specific application platform for students and instructors considering the problems mentioned below. A model, which can support students to comprehend main topics of programming and assist instructors during their teaching classes produced. Specialized learning tool, which named Algolyzer, developed to support basic concepts of learning algorithm and programming. Main contribution of the thesis is that this tool is independent of programming language, separates problems into smaller parts, and helps students do more practicing.

By using developed visual interface, students can create algorithmic steps to find a solution for a predefined algorithmic question. Developed tool gives students a chance to write code without diving into syntax errors, students only focus on the possible solutions. This contribution summarized that this educational tool helps students to improve their algorithmic-thinking abilities focusing on the solution.

Students consume their time and effort for PL syntax errors while developing programming codes. Creating the code in the selected programming language and displaying the whole code keep students more motivated with avoiding PL complexity. This is another advantage for students to achieve learning activity with a better environment.

In addition to this, Algolyzer is also a helper utility for instructors while teaching algorithms and programming. Having every action logs of students, instructors can have detailed information on how students use it, where they need more help, what

are the lacking parts in the teaching process. Final contribution of the thesis is that, evaluation of students' errors contributes instructors to explain misunderstanding parts more intensely in next the semesters within the related courses. In summary, this is a new approach on learning-teaching process of programming that usage of this developed tool contributes students and instructors with mentioned features.

1.3 Aims and the Scope

Depending on the Cormen, Leiserson, Rivest, & Stein (2009) definition "algorithm is a sequence of computational steps that transform the input into the output" (p. 1), aim of this study is aided students to produce an algorithmic solution for a given problem by separating the operational concerns and using a programming-language and environment independent method. The implemented application provides a visual interface that includes elements required to create the algorithm for a predefined problem. Since students only use visual elements and not any line of specific programming code, a generic code for the algorithm is being prepared in the background, and the programming code can be created in any programming language from generic programming elements prepared in the background.

The main idea of the study is to make studying algorithm and programming easier for the students. Being able to start an algorithmic solution for a problem without diving into programming language syntax helps starter-level students to be more effective and focused on the algorithmic thinking and solution domain. On the other hand, makes it easier to develop learning skills on basics of the programming.

With this study, it is aimed that;

- ✓ help students to understand basic algorithm and programming concepts
- ✓ provide a programming-language-independent tool to find a solution for predefined algorithmic question
- ✓ help students focus on implementing algorithmic steps with focusing only the program structure

- ✓ provide a solution for students so that they can do more practices with existing question repository
- ✓ save the actions of the students during their sessions
- ✓ get students usage statistics and evaluate students errors during programming
- ✓ help instructors to get more effective in-class sessions on lacking topics obtained from the evaluation
- ✓ create a platform and approach that can be used in both distance learning courses and in-class sessions

The developed tool targeted a user friendly, assistive, programming-language-independent, syntax free environment that helps students to focus on a single problem. Developed tool might have an important role for learning algorithm and programming with its features and contributions to overcome the mentioned difficulties.

1.4 Thesis Organization

In this chapter, we have stated that what we are trying to accomplish, what are our goals, and our contributions on this area. The rest of the thesis organized as follows.

Chapter two involves related works. Initially, general situation on this area is mentioned and similar, previous studies are introduced. Programming teaching tool, web based environments and some similar studies are summarized.

In chapter three, educational tools developed in the previous studies are explained and detailed. This chapter includes previous educational tools developed both on procedural programming paradigm and on object oriented programming paradigm. In addition to this, these mentioned solutions are brought together in a compared table.

Chapter four presents the design and implementation of new developed tool. During the thesis, a new solution, named Algolyzer, has been developed. General system of this developed tool and implementation details are introduced in this

chapter. The tool consists of various modules, which constitute the main body of the Algolyzer. These modules and structure details are in this chapter additively.

Chapter five includes the capabilities of the developed application. Position of the application in education process is mentioned in this chapter. Basic components and functional features within the capabilities of the study are explained. Moreover, this chapter includes previous version of this tool, which developed at the beginning of the study.

Chapter six focuses on the statistical results and evaluation of Algolyzer usage. Students' usage of this tool is cited in this chapter. Beside the general usage statistics, error rates and evaluation method are explained.

Chapter seven presents the conclusions, which includes the key contributions and fundamental findings of the thesis.

CHAPTER TWO

LEARNING MODEL & RELATED WORKS

Cognition and learning are main concepts in education. During years, many different researches have been achieved in this area. The literature identifies a variety of studies on learning theories and models as Bloom's taxonomy, constructivism, etc. Bloom's taxonomy is a method, which uses cognitive skills' categorizing, depends on the complexity order and there are six levels (Bloom, 1956). Constructivist learning theory depends on that learners not passively wait, actively construct the knowledge. Today constructivist-learning theory is the predominant paradigm in education. Concurrently these learning models and theories have been used on algorithm and programming learning process.

In general, studies started because of the importance and difficulties of teaching and learning algorithms in education process. Many studies with different models have been developed on teaching algorithms, programming, and data structures, which are not easy task for students. Some of these studies investigated the approaches in literature to teaching programming (Selby, 2011). Some of them suggest a new elementary programming education approach (Sajaniemi, & Hu, 2006). Study of (Marcelino, Gomes, Dimitrov, & Mendes, 2004) proposed an educational tool in constructivist perspective to help students.

2.1 Constructivism

Constructivism is a learning theory asserts that students construct knowledge combining the experiential world with existing cognitive structures rather than receive and store knowledge transmitted by the teacher (Ben-Ari, 1998). Learning depends on the active behavior of the students with what the student does, not what the teacher does. Most modern teachers shared that idea this form of constructivism is the best way (Biggs, 2003). For constructivist based programming instruction there are variety of activities as code walkthroughs, code reading, code debugging, and code authoring. Addition, these instructions include the code method bodies from

header declarations, or the use of rich development environments to support students for learning to program (Ben-Ari, M., 1998; Van Gorp, & Grisson, 2001). The study of (Wulf, 2005) is the application of constructivist pedagogical approaches to teaching computer programming in undergraduate courses.

The essential concept of the learning subject in question is emphasized in constructivist design (Nardi, 1996; Vygotsky, 1978). The providing student with the ability to represent and organize their knowledge is base role in the context of constructivist design (Jonassen, 1996). With this design, the role of appropriately-designed computer tools are crucial (Kordaki, 2010). Social learning theories emphasize the role of psychological tools and computer tools in the development of students' higher mental functions (Noss & Hoyles, 1996). Computer tools have been accepted as mind-tools, which can engage and support cognitive processing and critical thinking of learners (Jonassen, 1996). Social and constructivism learning theories are used in the proposed learning environment. Individual learning activities are realized on this study.

2.2 Related Works

Students need to spend their time to do practical activities to have the programming techniques. Students at beginning phase usually face difficulty associated with installing and using integrated development environment (Ng, & et al., 2005). Some studies, which will be outline, propose new applications and identify support tools to overcome these learning obstacles. Some of these studies proposed program development environment (Ziegler, & Crews, 1999). After improving applications, studies engage students to use these applications and usage activities often were examined (Jenkins, 1998; Hübscher-Younger, & Narayanan, 2003). Evaluation of some studies showed that support tools are effective (Costa , Aparicio, & Cordeiro, 2012). With these kinds of studies, there are some studies focused on the programming language selection in algorithm courses. For instance, the study (Chou, 2002) proposes students to use Python programming language in their classes for a significant time and detailed reports were generated. Beside studies on developing algorithm and programming skills, studies on different area such

improving web-programming skills were realized (Elgamal, & Abas, & Baladoh, 2013).

2.2.1 Introducing Algorithm and Programming with Didactic Scenarios

There are many difficulties for students who are the novice programmer face many mental obstacles in comprehending process of algorithm construction and programming functioning. This previous study depends on didactic scenarios that educational material organized. Didactic scenarios in this study include educational software to teach base topics of programming introduce students to basic programming principles and overcome difficulties for Secondary education (Dagdilelis, Satratzemi, & Evangelidis, 2004). Main parts of these didactic scenarios are the tool usefulness and the richness of interactivity as working in groups. Researchers of this study emphasized that didactical and pedagogical training of teachers in secondary education is important. Each new concept as loop statement corresponds some kind of problems suggested by teachers in developed software. Appropriate problem selection is important to get success on teaching new concepts to students.

With developing this educational tool and tool' usage in imparting education, findings of this previous study can be listed as;

- Formulating general rules associated with the significant didactic characteristics that should include in similar environment
- Educational applications is efficient just applications has framework based on didactic scenarios produced by instructor and supported by the tool
- Specific training, which give a chance to instructor for adaptation and usage in didactic scenarios, is essential for usage of these kind of educational application

2.2.2 ELP- Environment for Learning to Program

Study of Environment for Learning to Program (ELP) provides a web-based environment for teaching programming to students at Queensland University of Technology (Truong, Bancroft, & Roe, 2003). ELP depends on the “fill in the gap” style exercises. These exercises reduce the complexity for students in writing their programs. Students do programming exercises by “filling in the blanks” of a partial Java program in this previous study. The system compiles the completed program of students. If compilation is successful, system returns the resulting class to students in Java Archive format, otherwise messages with compilation errors returned to students.

2.2.3 Web-Based Environment Depending on Activities

Study of Ng & et al. (2005) proposes a web-based tool, which is an interactive environment for students to learn programming and for instructors to teach programming at distance learning. Students realize the programming practice and coursework on system depending on the basic functions of the programming. Students work on their programming codes in Java programming language without the complete programming environment and IDE. System gives appropriate feedbacks to students related their programming practices. Instructors follow students learning processes and compilation error messages. For the programming activities on the system, instructors generate and upload programming materials. Activity materials contain description file, template file and hint files, which includes a sample, output of the required program.

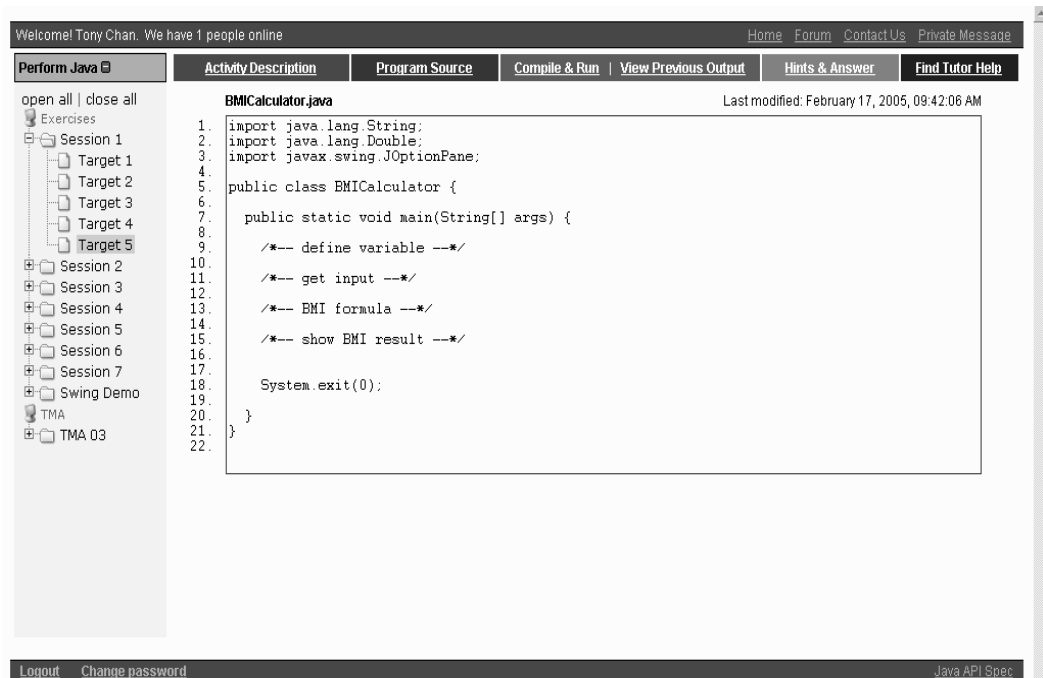


Figure 2.1 General view of application

In this related study; students work on the system depending on the activities uploaded by instructors and use template file shown in Figure 2.1 to generate their programming code on text editor. Students edit, compile and test programming code, if there is a compilation error, an error message send to the student. Otherwise, system generate executable file to the students. Error messages do not save, only students code file saved (Ng, & et al., 2005). Future works of this previous study was included collecting students' performance information to early detection of problems.

2.2.4 Programming Teaching Tool with Filling Fields

This previous study includes an experiment on development and testing a software tool for supporting of teaching introductory programming courses. Tool is a web-based application. Study depends on the spending time over on teaching the programming language syntax. When students make an effort for syntax, they throw other essential topics of programming as developing design skills (Al-Imamy, Alizadeh, & Nour, 2006). Features of this study were

- Templates generation depends on the main subjects,
- Helps students have different backgrounds to reach on an equal level,
- Allows passing different programming language,
- Usage as a web-based self-learning tool

Source file produced from instructor's template, which contains outline of the program structure. Students can generate their copies from templates with filling the required fields. For filling operation, students can delete statements and add new statement. Tool show the possible valid statements for required fields. Statements contains declaration, if statement, for loop, while statement. After completing the missing part, students can save developed program as C++ file.

For test' phase students of their institution used this tool. Examination results compared. Result of the study indicated that tool was effective on acceleration of learning programming language syntax.

2.2.5 Simplifying Algorithm Learning Using Serious Games

The study which name is Algorithm Visualization using Serious Games (AVuSG) includes visualization approach (Shabanah, & et al., 2009). This related study uses computer games to teaching-learning process of an algorithm. A Visualization approach has three different options as a text form, a flowchart form, and a game form. Learning theories integrates with game design with applying three learning models as Bloom Based, Gagne Based, and Constructivist Models in this study. This application has the user interacting level and the developer creating level for visualizations. Some algorithm visualizations, includes text, flowchart, and algorithm games prototypes were developed to validate the approach in this previous study.

CHAPTER THREE

EDUCATIONAL TOOLS

Introduction to programming is difficult mental task for students (Gomez-Albarra'n, 2005). Students face difficulties at beginning phase of the programming (Robins, Rountree, & Rountree, 2003). Various research studies performed on this area to support students for learning process. To compete with the difficulties of learning algorithms and to teach basic concepts of algorithm successfully, web-based, different graphical user interface based (Lazaridis, Samaras, & Sifaleras, 2010; Shakshuki, Kerren, & Müldner, 2007), abstraction-based, a specific algorithm (White, Martinez, & Rudolph, 2012) intended educational software applications have been implemented in the past. Suggestion an educational tool is one method used to deal with these complications in previous studies. Some studies depend on the evaluation of the specific educational tools used in class-sessions (Lazaridis, Samaras, & Sifaleras, 2010).

Graduate and under-graduate students in university education have used these developed tools and their experiences were followed in significant time. Study (Wang, Li, Feng, Jiang, & Liu, 2012) is one of these studies. Even, there have been some studies targeted secondary education students (Dagdilelis, & et al., 2004). These studies often are implemented as a web-based application. They provide students an editing, compiling, testing and debugging environment on the web for learning programming (Ng, & et al., 2005). Even some studies facilitate to visualize for program tracing process.

In general, to help students better understand and learn design and analysis of algorithms, algorithm courses include problems as programming assignments and exercises. Educational tool named AnimPascal study, tracks the actions of the students through a problem solution process (Satzemzi, Dagdilelis, & Evagelidis, 2001). Objective of these studies are to help students to understand the developing and other essential phases of programming and to aid teachers to discover the status of the students on the base of programming.

Studies, which support multiple programming languages, have been realized on the contrary studies based on the single programming language. That kind of studies assists students for improving their algorithms skills with using a base interface. In the same time, this gives a chance to implement their algorithms in various programming languages (Jain, & et al., 2010). There are some studies focused on teaching object oriented programming concepts. AEIOU is one of these studies which is software tool developed to support to understand of object oriented programming concepts (Licea, Juárez-Ramírez, Gaxiola, Aguilar, & Martí'nez, 2011). ObjectKarel study is another study, which provides a structure editor, runtime error detection, program animation and recording of students' actions (Satratzemi, Xinogalos, & Dagdilelis, 2003). ProTracer is different study that for visualizing students' program tracing processes (Chou, & Sun, 2010).

This kind of educational tools not replace the traditional methods of teaching, only complements the learning processes. Especially these developed tools support to students in practice and they include the foundations of programming. In general, these developed software applications allows students to study intuitively, graphically, and gradually. To develop new software these features are programming essential principles (Ferna'ndez, & Sa'nchez, 2004).

3.1 Educational Tool for Program Tracing:ProTracer

Program tracing is significant factor at learning programming phase especially novice programmers. This study designed a system which name is ProTracer, includes program-tracing processes for students and allows teachers to indicate students' errors in tracing execution. This study provides to students and teacher program tracing in view. Students can follow execution flow stepped and edit I/O display and variable values at each step and, they can modify tracing record. Tracing record indicates students' possible errors and misunderstanding of students. This suggest to teacher for future teaching. ProTracer has three interfaces; two of them for students which are trace programs and to preview their tracing records and another interface to correct students' tracing for teacher.

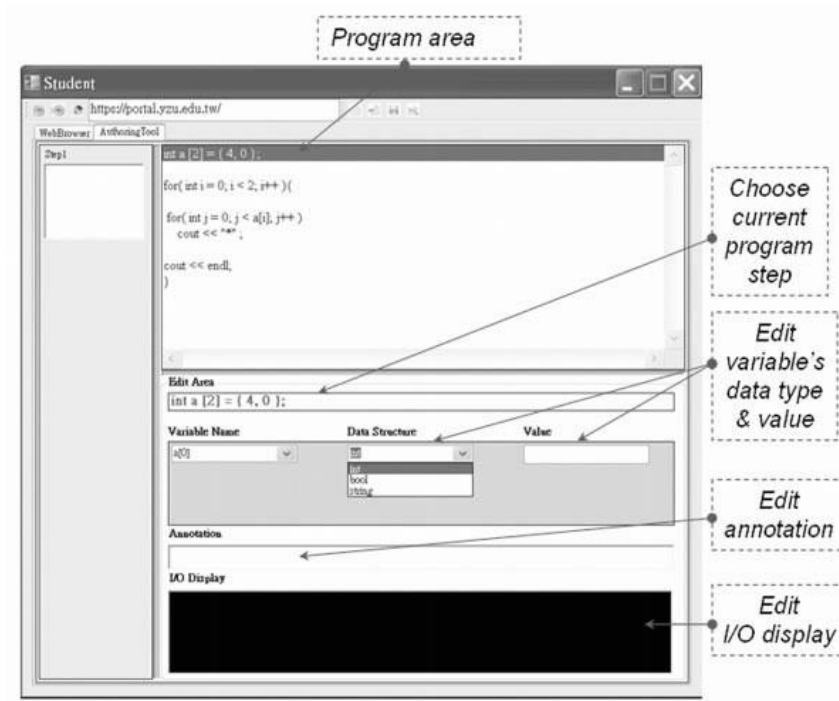


Figure 3.1 Program tracing interface of ProTracer

Engaging students to use ProTracer for program tracing in a Computer Programming II course and analyzing students' tracing records is an assessment of this study. The tracing records of 44 students collected and analyzed in this study. ProTracer result indicated that tracing abilities of students as poor and students' tracing errors were classified into execution step, variable, and I/O errors (Chou, & et al., 2010).

ProTracer supported program codes of variables, array, selection and repetition structure and this did not support functions and object-oriented programming. This study used files to store program codes and students' tracing records.

3.2 Educational Software: CGRAPHIC

CGRAPHIC is educational software, which was designed to learn the foundations of programming and C programming language. This study was developed in the Java programming language and executes on Internet or locally. CGRAPHIC provides theoretical and practical levels. There are several examples of the execution of

different exercises in the practical level and students can follow execution of program step-by-step or direct. This software includes and various parts as; a debugger of the C programming language, a programming online textbook, and a virtual tutor. Virtual tutor part of CGRAPHIC inside a graphical environment offers to student theoretical and practical learning of the basic concepts of programming. They implemented a set of graphical objects to develop new interactive exercises. CGRAPHIC provides support for variable, one dimension array, two dimension array, function, pointer, memory map, file, and structure objects.

This software involved main topics of a classical program in a first-year course in engineering studies and students in a first-year used this application. The main purpose of this study was to supplement teacher role with completing the traditional methods of teaching and develop teaching performance at programming base. There was English and Spanish version of this study. Spanish version was used at the University of Ma'laga in separate subjects as Foundations of Computing of the school of engineering, and Elements of Programming, and Practical of Programming in the school of computer science and as virtual tutor of a programming course in the Computing Virtual Services (Ferna'ndez, & et al., 2004).

3.3 Educational Tool for Understanding Algorithm Building

This study compound algorithms, base coding and multi programming language syntax in a single interface for learning algorithm and programming languages. Essential part of this study is that understanding of the main logic building and learning multi programming languages with a single interface. Tool was programmed in Python and contains script window, code generation module and sharing support. Sharing support part, which supports collaborative learning between students and teachers, realized with Remote Procedure Calls. At code generation drag-n-drop method used for image objects to learn programming languages majors. Each block images have text file depending on the programming language modules. When students select any block image, file content and necessary explanation related to selected programming language showed in script window (Jain, & et al., 2010).

3.4 Tools for OOP

Students in introduction to programming phase mostly meet difficulties independent of the programming paradigm as procedural or object oriented programming. In general, method selected by the researchers is identical even though there are some special kind of difficulties depending on the programming paradigm. Research activities often include development of programming environment to assist students to overcome these complications. Particular of these studies realized on object-oriented programming as ObjectKarel, AEIOU that mentioned in the next headings.

In the same time, there are also some studies, which developed for advanced programming concepts in object oriented programming courses. This developed tool support students to reach deeper level at programming knowledge (Licea, Juárez, Martí'nez, & Aguilar, 2008).

3.4.1 TeachingOOP: ObjectKarel

ObjectKarel is an integrated programming environment, which includes e-lessons series, special structure editor, and program animation to teach object-oriented programming paradigm. There were multiple ways to write program code in structure editor. Writing program realized in two ways. First is choosing the appropriate action like method declaration. Second is interacting with the system through dialog boxes. Students have three choices to execute program; running the program, tracing through the program and executing the program step-by-step (Satratzemi, et al., 2003).

Runtime error detection and recording students' actions were other properties of this developed software. Trial use of objectKarel by undergraduate students saved. For this, 20 undergraduate students from the department of Applied Informatics in Greece were used this environment. Students' programs and errors saved and evaluated in developed software concept. These give an idea to the teacher about students' problem solving techniques and errors. Teacher can follow students'

misunderstanding parts of object-oriented programming at beginning of the OOP learning.

3.4.2 Teaching OOP: AEIOU

In this study, development environment to help students for learning object-oriented programming with Java, which name AEIOU, developed. This tool, facilitated to students for programs' developments in programming courses. AEIOU includes three modules to support different type of students as novice, intermediate, and advanced (Licea, & et. al, 2011). AEIOU presents the project view with the classes' graphical representation and code view with the specific code. In the code view, students can edit, compile, and execute the class code. Environment gave class errors to the students during compilation. Displayed errors include translation in Spanish with more details. Base screen of AEIOU offers to the students various tabs to manage class operations for a deeper understanding of object-oriented programming concepts. AEIOU supplemented with ELVIA (Aispuro, & et al., 2012) Students in two Mexican engineering schools at University of Baja California and University of Sonora used this programming tool. Spending time of solving a problem, errors introduced by students and tool adaptation for other object-oriented programming languages are plans of this study.

3.5 Studies for Specific Algorithms

Some studies propose learning tools for teaching of specific algorithms such as genetic algorithm, ant colony optimization (Li, & Liu, 2009), etc. Some studies depend on the learning sorting algorithms (Kordaki, Miatidis, & Kapsampelis, 2008). Certain of them depends on the new approaches such Reinforcement Programming that used to generate sorting algorithm (White, & et al., 2012). (Byrne, Catrambone, & Staskoc, 1999) conducted the using animation for depth first search algorithm for students learning.

3.5.1 Learning of Sorting Algorithms

A web-based environment was designed to support secondary level education's students for learning of sorting algorithms and pilot evaluation of environment was presented in this study. The environment design based on the modeling methodology, considering modern constructivist and social theories of learning. Developed sorting environment also give a chance to students for typical sorting algorithms' learning as Bubble-sort, Quick-sort and Selection-sort. Data of pilot evaluation study of sorting environment was analyzed and these results are obtained: students used all the representation systems and they found environment attractive and easy to use (Kordaki, & etc., 2008)

3.5.2 Learning Tool of Genetic Algorithm

This study suggested a learning tool to teach and study the genetic algorithm. The user of the tool can study and manipulate the algorithm easily with friendly graphical user interfaces. This presents that the aim of the developed tool is supporting to teach the algorithm. Tool interface includes Genetic Algorithm parameters area, and computation processes area. Graduate students at Nanjing Agricultural University get a chance to use this tool to learn the genetic algorithm (Li, & Zhang, 2010).

3.6 Evaluation and Comparison of Tools

In this chapter, educational tools in literature are investigated and the general features mentioned in previous headings are summarized. Table 3.1 includes the properties of some educational tools used to algorithm and programming learning in previous studies. There are more similar studies in literature, this table not include all of these studies in this area. Studies are specially examined considering the error detection as saving students errors, giving appropriate feedbacks to students related to their errors.

Table 3.1 Features of previous educational tools in literature

Name	CGRAPHIC	ProTracer	ObjectKarel	AEIOU	AVuSG
Multi PL Support	Programming in C, but can be adapted to another PL	Only C++	Object Oriented Programming	Java	Non-existence
Students Usage	Non-available usage value, the school of engineering, school of computer science	in Computer Programming II course	Applied Informatics department	in two engineering schools	No information
Save Usage-Errors	No, only informing	Marked by instructor	Only informing	Non-existence	Non-existence
Error Message	Nothing	Tracing Error message	Run time error detection	Compile errors	Absence
Graphical Support	Graphical objects for program execution	Students tracing, teachers correctness interfaces	Explanatory visualization, highlighted, show object messages	Project view, code view	Text, flowchart, game forms
Application Type	Web based	Web based	Not certain	Desktop application	Desktop application
Topic Concept(main/advanced)	Base topics, advanced (arrays, function, memory map, etc.)	Base topics	Base topics	Object Oriented Programming	Specific algorithm & data structure
Process Type (tracing, code generation, show sample code)	Follow exercises with graphical objects & code part programming content	Program Tracing	Editor with choices of class, object operations, execution of programs, e-lessons	Code generation	Using games

Considering these general features of some previous educational tools, a new software solution can be developed for learning algorithm and programming. In this study, a new educational tool is proposed, which aids to students to learn basic concept of algorithm and programming in the first phase of the work. In the second phase of the study, graduate students at Dokuz Eylül University are engaged to use developed tool. The students' usages of new developed tool–Algolyzer are investigated at the end of the study.

CHAPTER FOUR

DESIGN AND IMPLEMENTATION OF NEW INTRODUCED EDUCATIONAL TOOL: ALGOLYZER

In this study, a new educational support tool has been developed to help students for learning basics of programming and to assist instructors during teaching classes. An educational tool named Algolyzer -the visual interface tool created for this model- provides a programming-language-independent environment for students and gives information about the misconception topics in the teaching process for instructors. In this chapter, developed tool will be explained in details. During the study, an appropriate model for supporting programming learning produced, depending on the investigation of the previous studies and observation of the algorithm and programming courses in computer engineering department at Dokuz Eylul University (DEU).

At implementation phase of the study, general system architecture designed firstly based on the requirements and plans. In second step sample problem repository has been created and the possible programming operations which should be in the application were determined. After all decisional process, implementation of the application was began and completed. Completed application provides different opportunities for instructors and students. This software provides a web based access to both instructors and students. Students can create their own projects on the tool. During these activities, usage errors of students are saved automatically and can be analyzed for detecting the main topics of programming, which is not being understood properly.

4.1 General System Design

The tool developed as a web application. Tool interface provides multiple usage scenarios both in-class sessions and distance learning systems as the previous version of this study (Yıldırım, & Kut, 2010). General process of application depends on

focusing on understanding and solving a single problem. Programming can be divided into four steps (Winslow, 1996):

- understanding of the problem
- definition of the problem solution initially in any form, such as text-based or math-based and in a computer compatible form
- create solution using selected programming language
- testing and debugging of the solution program

Students firstly try to understand problem given by the system, and work on problem and generate solution, then student passes to next problem. At the beginning, students determine main operations with solution separation. This method supports the students to improve an algorithm easily. The structure of the system can be summarized as giving problem to students and saving errors made by students in problem solving phase illustrated in figure 4.1.

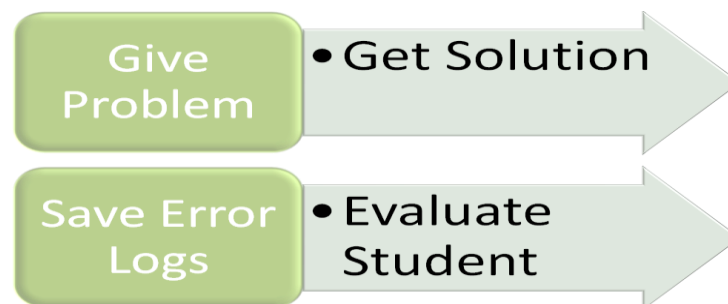


Figure 4.1 Main operations of general system

After finalizing the first usable version, students of computer engineering department at DEU used the application. During this phase, we kept the system updated and made minor changes depending on the feedbacks to get experiences that are more efficient. In the last phase of the study, we investigated the system usage and evaluated the collected actions, operations and errors data, which will be mentioned in the next chapters.

4.1.1 Operational Level of System

Algolyzer is a web-based application and has two parts, which are the user and system parts as demonstrated in Figure 1. User sends code pieces of operations to the system part. Code generation is done using the selected programming language depending on the algorithm that user created using visual elements of the application.

Error detection runs on system part. If there are any errors in user code part, appropriate messages send to the user. During the session, user actions and errors are saved to the application database and they are kept notified about any problems in the algorithmic rules. Saving every action and any possible errors to log database helps instructors and analyzers to investigate and determine the lack of topics. Briefly, the system side responsible for handing the requests from the users, keeping user logs and serving them as a shared compiler and giving user create and download code file options.

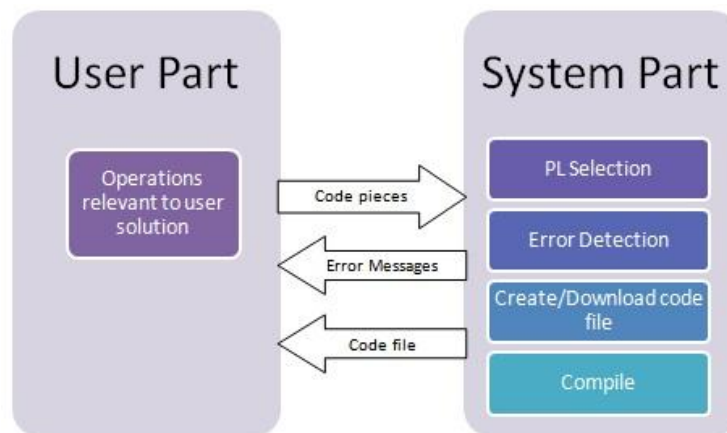


Figure 4.2 General system views on operational level

4.1.2 Infrastructural Level of System

On infrastructural level, frontend application and test module are at the top of the layered model. Domain model and processing library are over the data access part. The last part that Algolyzer builds on, .NET Framework is bottom of the structural schema. All of these modules implemented in Algolyzer framework can be considered from Figure 4.2.

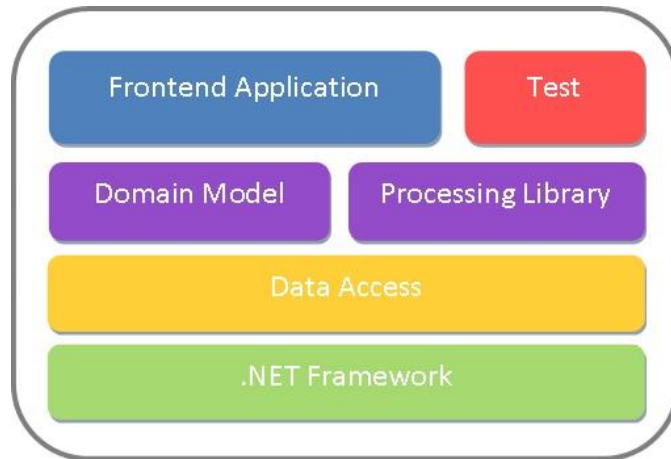


Figure 4.3 General system frameworks

Overall classes associated with the infrastructure can be observed from Appendix A. These classes perform the various operations and statements of developed tool.

4.2 Library Module of Algolyzer

4.2.1 Processor

Processor module includes classes presented as class diagram in figure 4.3. Two different programming language processors were created in this phase of the study and can be extended with more programming languages.

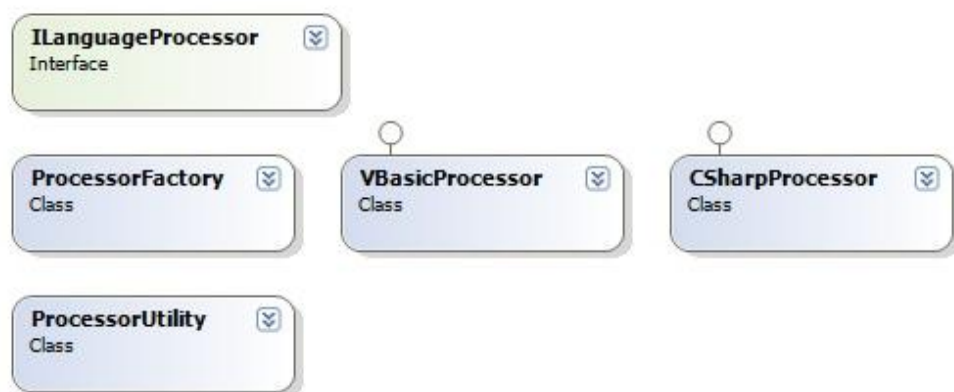


Figure 4.4 Class diagram of processor

A generic base structure has been created and applied for multi programming languages support. To get appropriate operation a Processor factory class was developed. Code is generated depending on the selected language with responsible processor.

```
Processor Factory
Function Get Processor(programming language pl)
    case pl of
        ProgrammingLanguage.VBasic: return new VBasicProcessor()
        ...
        default: return new CSharpProcessor()
    end case
endfunction
```

4.2.2 Services

Services are the main contact points between user interface layer and data layer. Two most important services being used widely during the application are, error log service and error message service.

Validators are based on the statements and control and validate the operations. They all implements a shared interface named IValidator and located in services section with other two service shown in Figure 4.4.



Figure 4.5 Services of Algolyzer

4.2.3 Statements

Statements developed depending on the main operations of programming. Statement classes correspond to the operations on the user interface in backend codebase. These are condition, read/write, if/else, assignment, loop, variable statements illustrated in Figure 4.5. All statements are derived from IStatement interface.

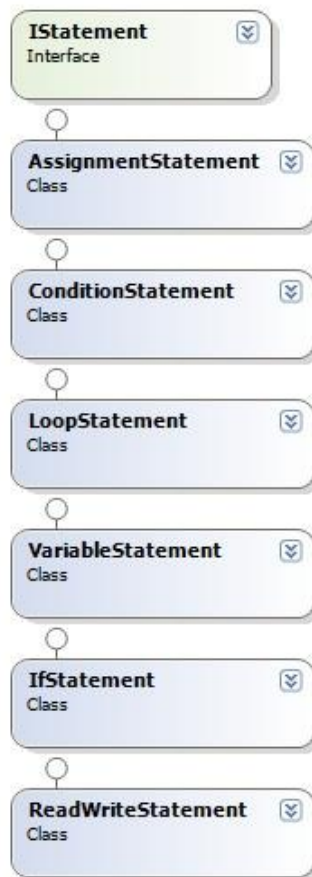


Figure 4.6 Statement classes

4.3 Data Module

Data module includes the database operations. General classes of data module illustrated in Figure 4.5. Operation, problem, error type, user log, error log, problem text, error text and language used are main types of data module. For a problem operation, which supports multilingualism, problem, text and language objects are referenced.

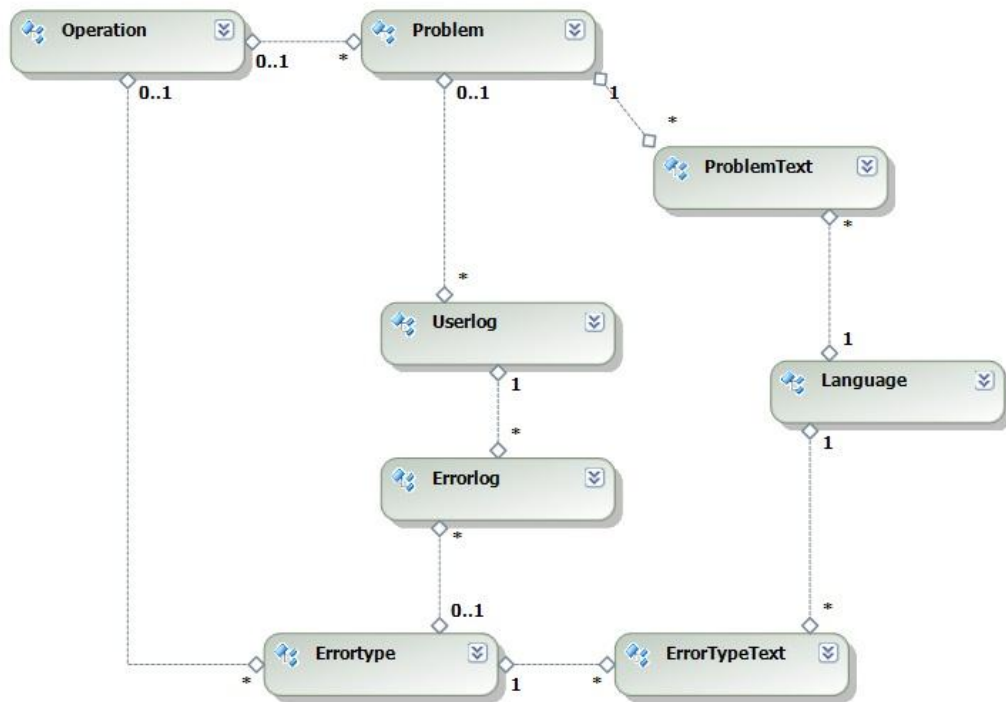


Figure 4.7 Data class diagram

4.4 Code Generation

A processor is created depending on the selected programming language primarily on code generation phase. Statement's creation is implemented in the following phase. Figure 4.1 illustrates the creation of the statements in processors. Parts of code creation were implemented in the code generation method, which includes the generation methods of all statements. Statements are common for all languages. Nevertheless, the generation is different for each of the programming languages.

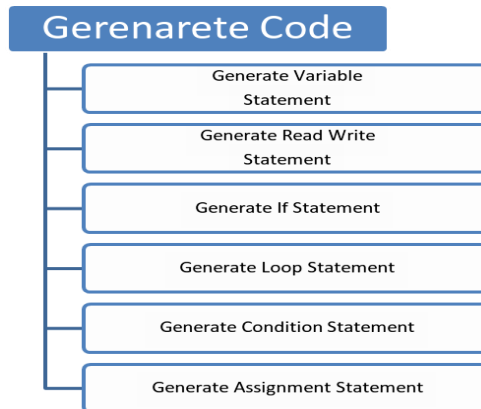


Figure 4.8 Generate code of processor

4.5 Statement Details

Read/write statement includes operation type and variable statement. If statement has if type, condition statement and inner statements as read/write, assignment, etc. For all language in code generation, every statement and its fields are common. Assignment statement includes the left side variable, mathematical operator, first right side variable and second right side variable in the following code block.

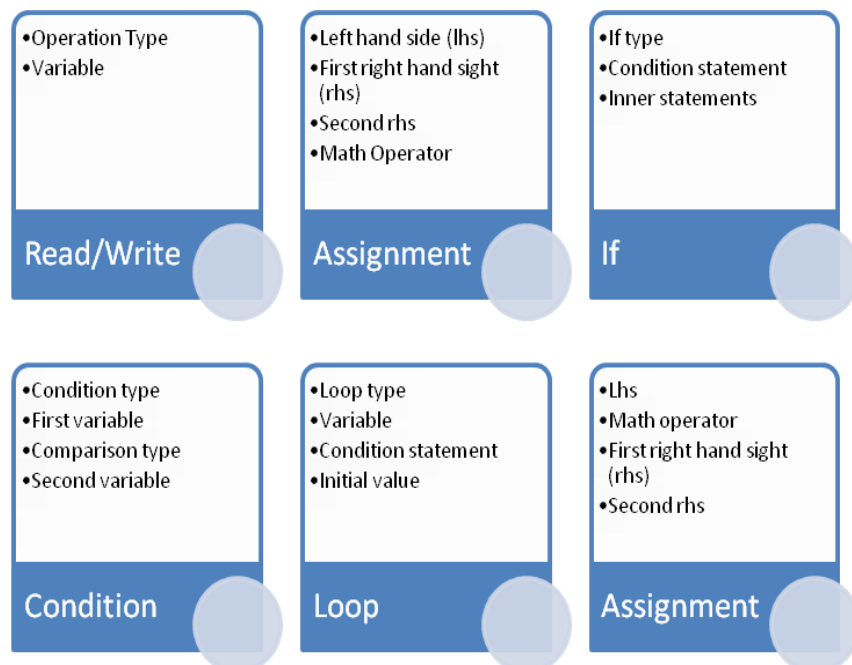


Figure 4.9 Statements and subfields

4.6 User Interface

Developed tool functionalities mentioned previous headings, are being used by students through a visual interface. Figure 4.9 presents the general view of the interface. Detail screens of the application are in Appendix B. There is an instance of one scenario.

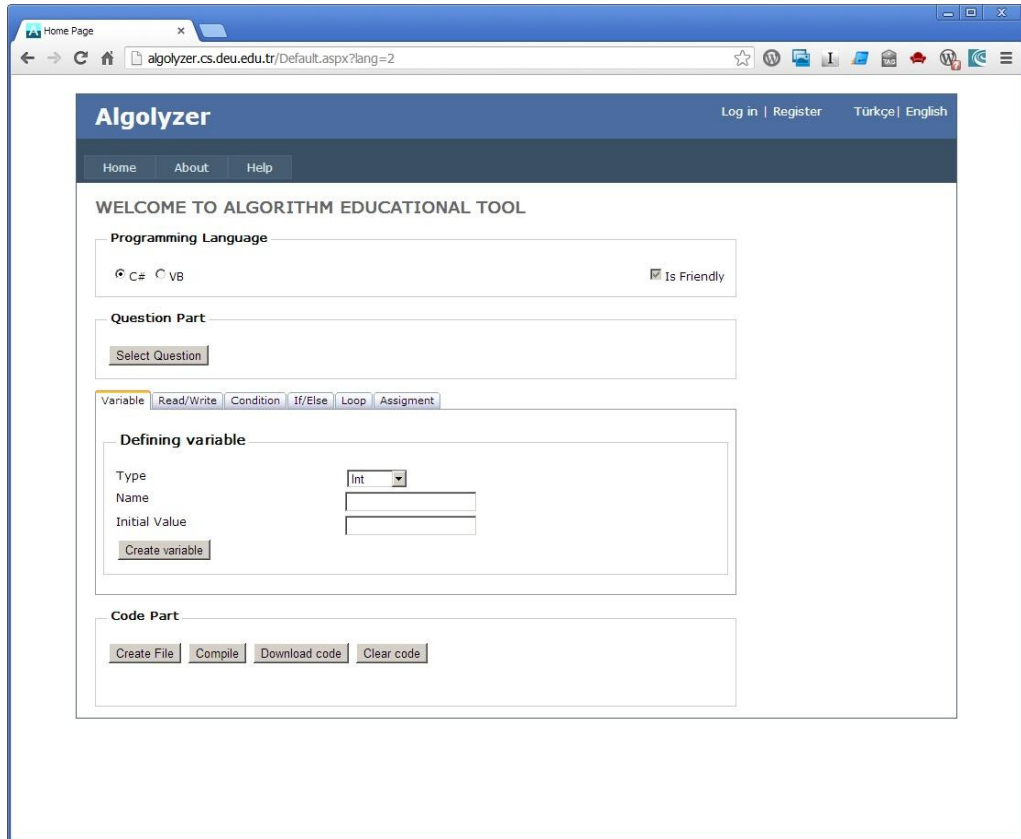


Figure 4.10 General view of developed tool

CHAPTER FIVE

CAPABILITIES OF ALGOLYZER

An algorithm and programming course introduce students to programming. In this study when determining tool's capabilities, computer-engineering students' thoughts were considered. A survey on difficulties of algorithm and programming learning were prepared to get benefit during tool development phase. Students who are failed and face difficulties in algorithm and programming course participated to survey. 26 first year students were selected randomly. Feedbacks of the students on difficulties of learning programming are evaluated. All 26 students marked at least one difficulty on programming learning process. Passing the abstract thinking had the highest value, difficulties caused by programming languages and understanding of the problem exactly are the substantial ratios detailed in Figure 5.1.

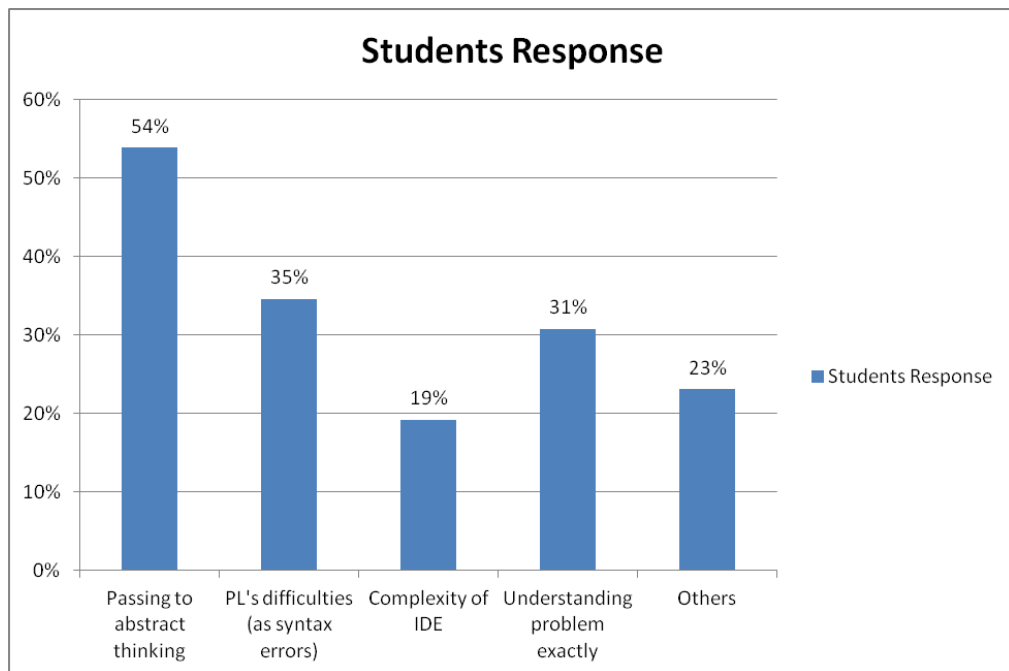


Figure 5.1 Ratios of difficulties in programming

Students can use the developed tool with its feature set. Proposed tool has an easy to use visual interface and component based environment. During this study, various versions were improved. First version of this study was a desktop application. The next, web-based version of the developed tool consists of the following basic components and functional capabilities.

- Problem part, operation part, and code part in user interface
- Multi programming language support
- Tool multilingualism support
- Error detection and validation

These capabilities of Algolyzer mentioned above will be explained in details in the following headings.

5.1 Developed Tool Position in Learning Process

In traditional education, instructors transmit knowledge to the students and students try to store knowledge in mind. When students are elements of the learning process as student-centered models, these proposed systems facilitate students to store knowledge. In Algolyzer, students can do more practices on programming with using developed tool and get feedbacks about errors in Figure 5.2.

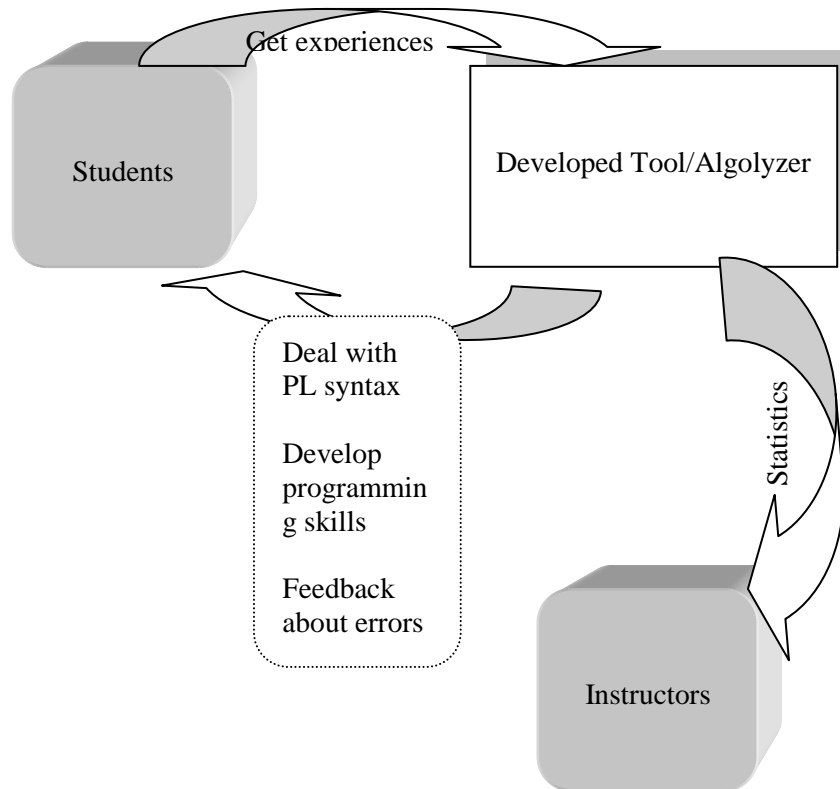


Figure 5.2 Developed tool position in learning process

5.2 Base Components of Interface

The developed tool has three parts in its visual interface; problem, operation and code sections. The workflow of these base components illustrated in Figure 5.3.

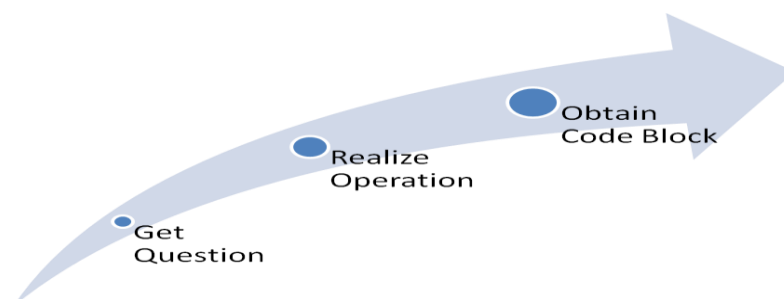


Figure 5.3 Workflow of the base components

Firstly, problem part is coming to students; then student begin to create solution for the problem using the tool interface. In solution phase, the students navigate through related tabs of the tool to create operations. Each student should create

his/her own solution in a syntax free way. The tool depending on the user's solutions in the code part will automatically generate program code of a given problem. The mentioned general workflow can be followed from graphical environment in Figure 5.4.

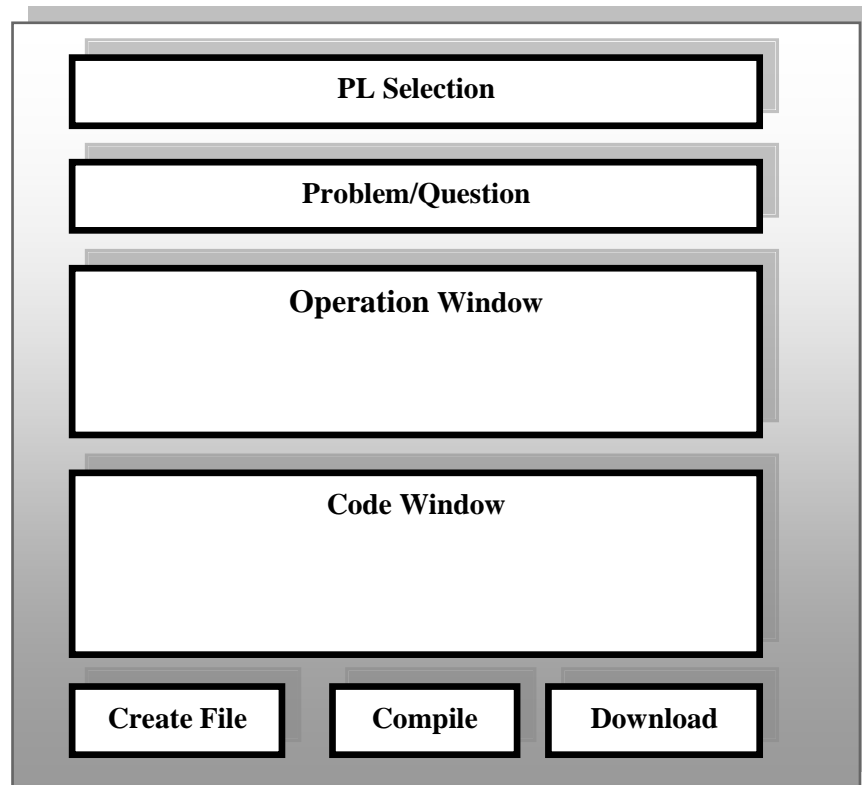


Figure 5.4 General view of GUI

5.2.1 Problem Part of the Developed Tool

Question/problem repository was created and problems were categorized depending on the subject groups (operations) and difficulty levels illustrated on the Table5.1 for problem component. Problems appear randomly from the repository to the student side when the student clicks select question button from the visual interface. Focusing on understanding and solving a single problem is important. At any time studying on understanding, solving and testing of a problem should be completed by students then next question will take place (Müldner, & et al. 2008).

Table 5.1 Problem repository of the system

ProblemID	ProblemText	Problem Degree	Operation ID
1	Write to screen welcome to programming	1	2
2	Take user name and age then display this information on the screen	1	1
3	Find the total of given two numbers	1	1
4	Multiply two given number and display result on the screen	1	1
5	Write days of week on the screen according to given number(1-monday,..)	1	2
6	Find the average of midterm and final grades given by user	2	6
7	Convert a measurement given in km to the equivalent number of m, cm and mm	2	6
8	Find triangle's surroundings given three numbers	2	6
9	Find rectangular area depends on the a,b edge values	2	6
10	Find rectangular surroundings depends on the a,b edge values	2	6
11	Tell the given number is positive or negative	3	4
12	Tell the given number is odd or even	3	4
13	Print a given digit(0-9) in text (1-one,2-two,...)	3	4
....
26	Find the total of n numbers given by user	4	5
27	Draw rectangular on the screen with #	4	5
28	Draw diamond on the screen with *	4	5
29	Read one character from user and display this character on the screen	1	2
30	Take two number from the user and display subtraction result on the screen	1	6
31	Take x,y from the user and calculate power(x,y) using loop and display result on the screen	4	5
32	Find the biggest number given two numbers taken by user	3	4
33	Calculate and display absolute value of number given by user	3	4
34	Take midterm,assignment and final grades from the user and calculate average of course(%20Ass,%30mid,%50final)	2	6
35	There is two nested circle,calculate the difference area of these two circles according to R1,R2 value	2	6
36	Take course grade from user and display passed or failed condition depends on the 70 point	3	4
37	Write a program that print the months of a given season	3	4

Table 5.1 Problem repository of the system (cont.)

38	Take number from the user between 1-1000 and adds all the digits in the number(562 - 13)	2	6
39	Write numbers between 4 to 96 (4 10 16 22 ...) using for loop	4	5
40	Write numbers muplications of 3 from 1- 100	4	5
41	Find the second largest number of given 7 numbers by user	4	5
....

5.2.2 Operation Part of the Developed Tool

Achieving a well-planned learning path is important at teaching-learning process. In this study, interface separates problems into smaller parts as operations. Operations are determined depending on the main subjects of algorithms and programming. There are six statement types; variables, loop, if-else, read-write, assignment, and conditions as illustrated in Table 5.2.

Table 5.2 Operation table

OperationID	Operation Name
1	Variable
2	ReadWrite
3	Condition
4	IfElse
5	Loop
6	Assignment

Each operation has its own child fields, which explained in details in previous chapter. In this way, students can learn steps of operations during development phase. Thus, they also have a chance to do more practice on main and subfields of programming topics. Programming-language-independent feature embedded in this part. Using operation parts to create solution algorithm is important for students since they generate a syntax-free code. Students can focus on algorithm design. Thus, they are more motivated and have more chance to improve their programming skills.

5.2.3 Code Part of the Developed Tool

Algolyzer generates source code automatically depending on the actions of the students in operation part of environment after completing the error checks. At each step of the solution, students can follow their algorithm in code part of the interface on selected programming language. There is a “clear” option to clear all generated code and start the problem solution over again.

In the code part, there are also three actions can be taken about the solution code. Students can create source code file and download this generated file wherever they want to save. In addition to this, students can compile their solution on the tool interface to get more realistic compiler warnings.

5.3 Multi Programming Language Support

When students make an effort for programming language syntax, they spend much time and there is a little time to improve their programming skills (Al-Imamy S., et al., 2006). Programming-language-independent code generation is important and this feature is one of the main study objectives.

Generic algorithm elements developed in the backend, which allows generating code in any programming language. For multi-programming-language support, this generic structure is being used through Processor factory mentioned in previous chapter. Solution source code is generated in the selected languages depending on the operations and orders of student’s operations.

5.4 Language Support

Turkish language support on interface provided to students besides English. Especially, giving error messages in Turkish supports students to understand midpoints more clearly. Certain previous studies indicate that students at the beginning phase of programming have problems about understanding messages of IDE or any programming language compilers (Licea, & et al., 2011). Getting error messages in details and in native language is more beneficial for students.

5.5 Error Detection

In general, compilers are designed for advance programmers, not for novices (Satratzemi, & et al., 2003). Students spend their much time for debugging operations, sometimes they do not understand error message while improving their algorithm solution. In this study, error detection is over the debugger shown in Figure 5.5.

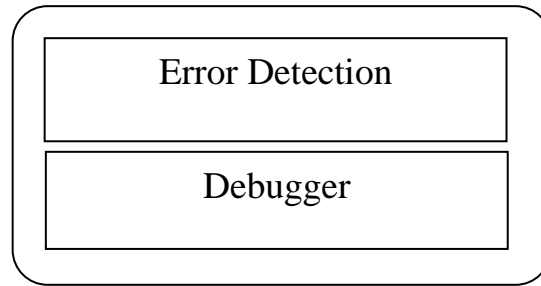


Figure 5.5 Error detection mechanisms

Error detection of this system is processed before debugger. An appropriate warning and errors messages delivered to the students by the system depending on the operations of the student. Programming-language syntax errors and some logical errors that caught by the developed tool identified in Table 5.3. Errors are separated with the operation type and error degree. Error degree indicates that if the error type is a syntax error or logical error. Third level states the certain logic errors.

Table 5.3 Types of errors made by students during the programming

Error Type ID	Error Type Name	Operation ID	Error Degree
1	VarIncomplete	1	1
2	VarTypeMismatch	1	2
3	VarNaming	1	3
4	VarDefineAgain	1	2
5	RWIncomplete	2	2
6	RWConvert	2	2
7	RWParameterMissing	2	1
8	CondIncomplete	3	2
9	CondIncompleteMulti	3	2
10	CondTypeMismatch	3	2
11	CondSameVariable	3	2

Table 5.3 Types of errors made by students during the programming(cont.)

12	CondSameConditionMulti	3	3
13	CondBoolCtrl	3	3
14	IfIncomplete	4	2
15	IfIncompleteElse	4	2
16	Command missing	4	3
17	LoopVariableInitialMissing	5	2
18	LoopConditionMissing	5	2
19	LoopIncDecValueMissing	5	2
20	LoopCommandMissing	5	2
21	LoopLogicError	5	3
22	AssignmentIncomplete	6	2
23	AssignmentTypeMismatch	6	2
24	AssignmentNumericTypeMismatch	6	3
25	AssignmentWrongStringOperator	6	3

System executes the error detection for students on each operation steps. If there is any error, understandable error messages are reported to the students as illustrated in Figure 5.6.

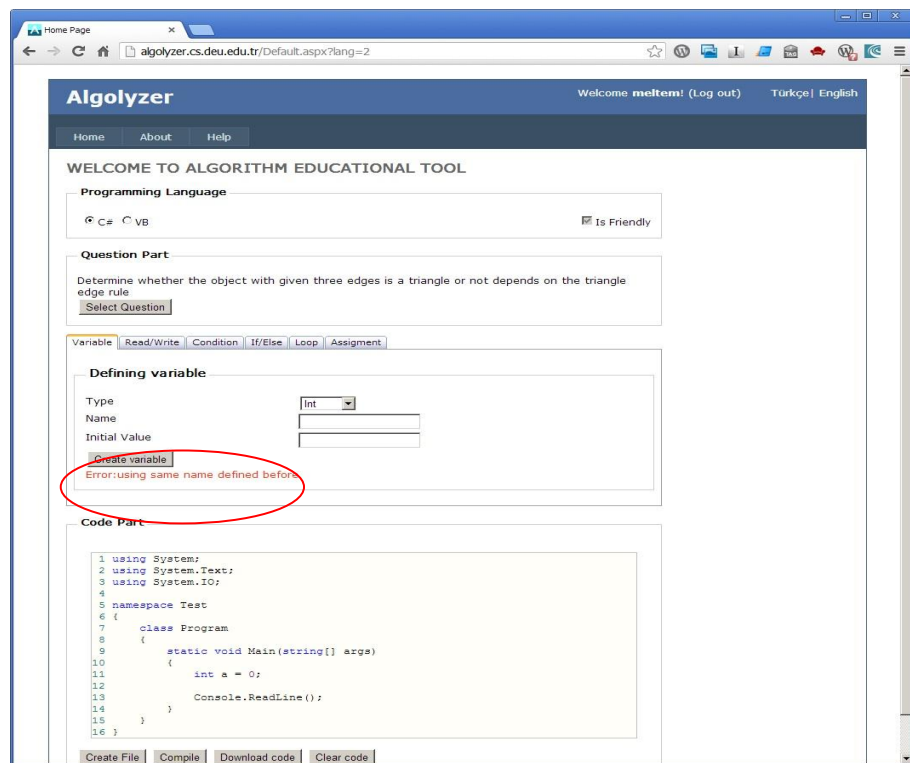


Figure 5.6 Sample screen of error detection

The application saves these activities during sessions of the students. The instructors can examine the usage reports and try to get more effective in-class sessions. Instructors investigate the logged actions and error logs. Thus, getting statistics from the students' logs can provide a possibility to determine misconceptions. During algorithm and programming courses, instructors can consider errors on main topics and subfields, which shows misunderstanding parts for students.

5.6 The First Version of the Developed Tool

The previous version of this study implemented as a desktop application. There are some different features besides similar features. On operational level, application has the same capabilities as problem, operations and code parts while developing solution of the problem. Different capability of these features, there is a flowchart support capability. Students can follow operations on flowchart and when the students complete the problem solution, overall student's solution workflow can be obtained in the application interface as illustrated in Figure 5.7. Creating executable file and running this created file option are other different parts of this version. These different capabilities actualized easily associated with the application type. In detailed, more instance screens of the previous version of application hold in Appendix C part.

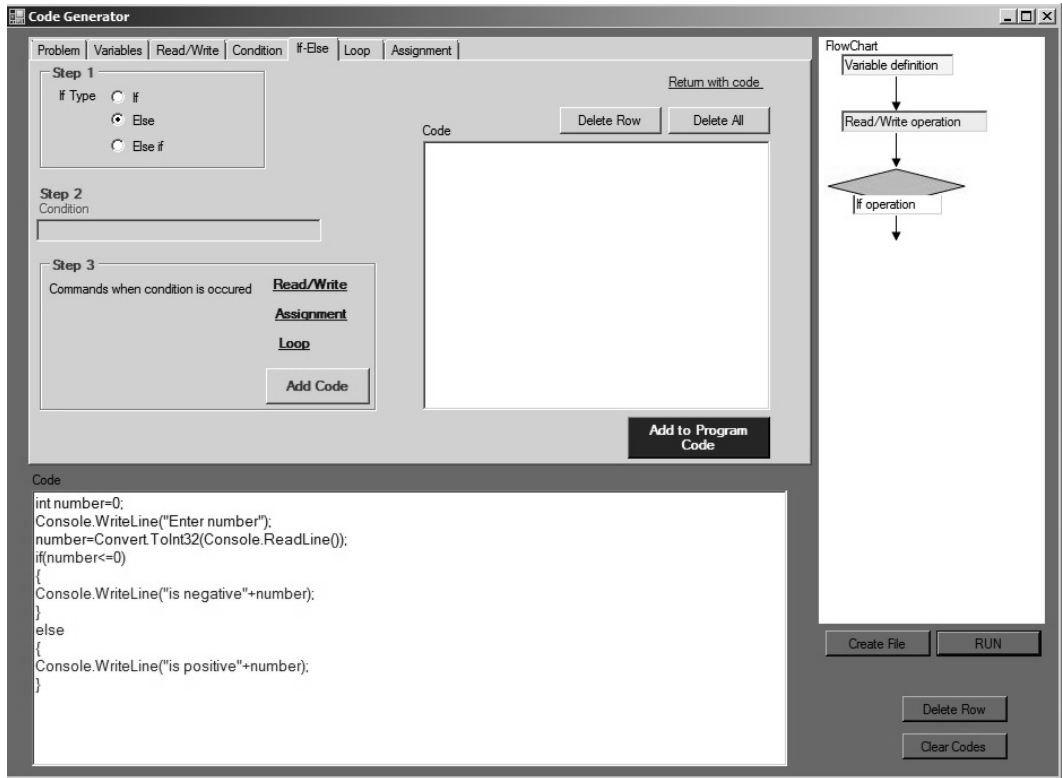


Figure 5.7 An overview of previous version of application

CHAPTER SIX

RESULT AND EVALUATION OF ALGOLYZER USAGE

In imparting education, to have an idea on misunderstanding parts of the courses is essential for instructors. Particularly at mental and abstract courses as an algorithm and programming courses, learning is difficult for students. Students have misconceptions, related with facing problems on introduction phase of algorithm and programming. In our study, to have an idea related to these complications, developed tool tracks every action of students and collects the errors from the compilation results and higher-level application validation rules. In this way, instructors can follow the students' learning process and their misconceptions associated with programming topics.

For study's assessment, developed tool was published in <http://algolyzer.cs.deu.edu.tr> address and we engage students to use this educational tool. Students of computer engineering and computer programming department of Dokuz Eylül University were informed about Algolyzer and application published web site address. Students used developed tool within the scope of their algorithm and programming language courses. After students' usage, application saved students' activities and errors. Then we investigated and evaluated those knowledge concerns on the usage and errors made by students. In this chapter, this investigation and evaluation will be presented and explained in details. With this evaluation, the instructors have usage reports and they tried to get more effective in-class sessions during their algorithm and programming courses.

6.1 Engage Students to Use Algolyzer

Practices of courses are too important for students when mental concerns are matter of the learning process. Abstract knowledge can be followed in tangible forms with similar applications as our developed tool. In this way comprehension of those topics can be facilitated for students. Beginning from this point, we engage our students to use developed tool- Algolyzer.

The developed application was announced to our starter-level class students at computer engineering and computer programming department of Dokuz Eylül University. Students used developed tool published in <http://algolyzer.cs.deu.edu.tr> address during the one semester within the scope of their related courses. There are two different experiment group illustrated in Table 6.1. Students in first study group used this tool in algorithm and programming course. Students at second study group used this tool in programming languages course. Each study was realized on separate semesters with different students.

Table 6.1 Experimental groups of study assessment

	Department	Class	Course name	Code
First study	Computer Engineering	Starter- level	Algorithm &Programming I	Study A
Second study	Computer Programming	Starter- level	Programming Languages I	Study B

System saved students' activities and errors during their sessions. At the end of the semester, logs obtained from the system database were investigated.

6.2 General Statistics

The login/registration, operations and errors made by the students were saved to system database when students use the application. This gives an idea related to students sessions, their usage and erroneous state while improving their solutions on the tool. Thus, erroneous statistics provide the opportunity for the instructors to determine lacks on topics.

The data retrieved from the user experiments, values in Table 6.2 were obtained for study A group. Investigation can be summarized as; system collected 84 student's registration and 950 sessions. Session corresponds to the students' login and chooses a question. The questions have been chosen by students are randomly selected from database. In this group, not all students made an error at the problem solution phase and compiled their improved algorithm. Solution code compilation was not succeeded in general. Record of compilation means that students select one question and perform to complete his/her solution and then compile code file.

Table 6.2 General Statistics of System Usage for Study A

Record Name	Count
Total number of students	84
Total number of sessions	950
Total number of erroneous sessions	80
Number of questions	52
Number of questions error(s) made on	32
Number of error types in erroneous sessions	15
Number of operand/statement types	6
Number of compilation	39

General statistics depending on the usage of the students in study B group were listed in Table 6.3. The 42 students used the tool related to their courses. Session number is a little low, when these values compared with study A group's values. Another remarkable state is number of compilations is too few. This means that most of students cannot come to the end of their solution or not prefer to compile.

Table 6.3 General Statistics of System Usage for Study B

Record Name	Count
Total number of students	42
Total number of sessions	665
Total number of erroneous sessions	29
Number of questions	60

Table 6.3 General Statistics of System Usage for Study B (cont.)

Number of questions error(s) made on	18
Number of error types in erroneous sessions	9
Number of operand/statement types	6
Number of compilation	2

6.3 Distributions of Students Errors During Programming

In problem solving phase, students make an error associated with their algorithm and programming. System saved these errors during the students' session. At the end of the semester, we worked on these errors data. In this part, types and distributions of errors made by students explained in details. With these distributions and evaluations, topics of algorithm and programming course, which not properly understood, can be determined.

6.3.1 Error Distributions on Base Topics

In study A, there are 80 errors made by students while generating algorithmic solution of the problem using the main operations. Error type and their distribution illustrated in Figure 6.1.

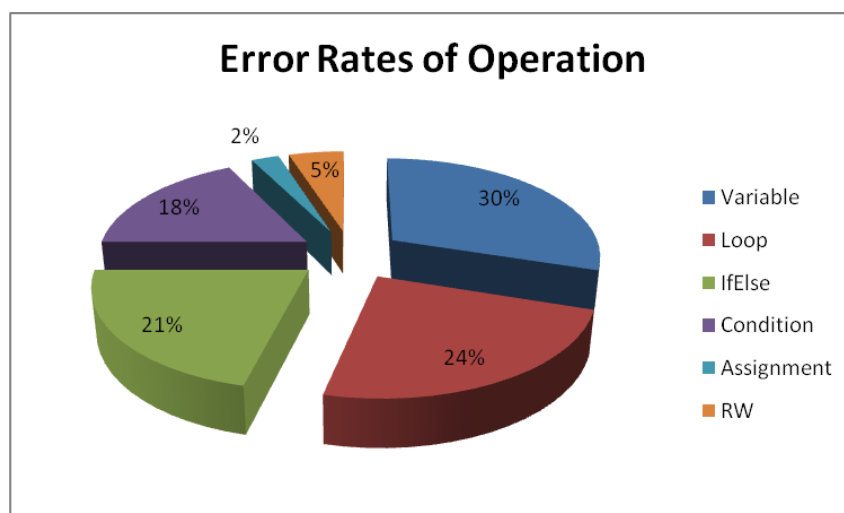


Figure 6.1 Error rates of base operations for Study A

For Study B there are 29 errors made by students during their solutions. Figure 6.2 shows the distribution of the main operation errors. Operation that has the highest distribution is the same in study A. Other error types are changing on following rates.

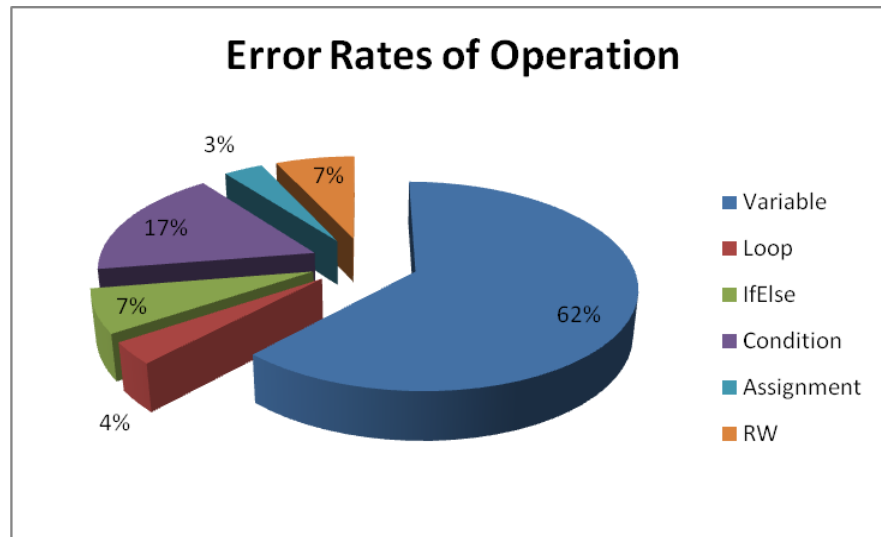


Figure 6.2 Error rates of base operations for Study B

6.3.2 Error Distributions on Subfield Topics

System saves errors associated with subfields of the main topics when students are working on the problem solution. For instance, student made an error on loop operation, error part as variable or condition parts of loop statement saved to database such as condition lacking or command missing in loop statement. Order of error distribution in subfield topics are changed a little when compared with the results of main operations in previous heading. On the contrary, in previous ratio, the most errors comes from “If” operation of all main subjects. This show that students made incomplete if/else statements while creating their algorithm. Some specific variable, loop and condition errors with the same ratio value pursues highest topic. Defining same variable, condition missing of loop, using same variable in condition are the second high-rated errors made by students. Other subfield errors are figured in Table 6.4 for study A.

Table 6.4 Subfield error rates of Study A

ErrorType	Operation	Rates
14	IfIncomplete	18,8
4	VariableDefineAgain	11,3
18	LoopConditionMissing	11,3
11	CondSameVariable	11,3
2	VariableTypeMismatch	10,0
17	LoopVariableInitialMissing	8,8
1	VariableIncomplete	8,8
13	CondBoolControl	5,0
5	RWIncomplete	3,8
19	LoopIncDecValueMissing	3,8
15	IfIncompleteElse	2,5
7	RWParameterMissing	1,3
8	CondIncomplete	1,3
22	AssignmentIncomplete	1,3
23	AssignmentTypeMismatch	1,3

Distribution of the subfield errors for study A in Figure 6.3. Error type, which has the highest value based on the subfield different from the most intensity of error type one based on the main subjects. When we examine the different errors, main subject variable and loop subject have the highest intensity.

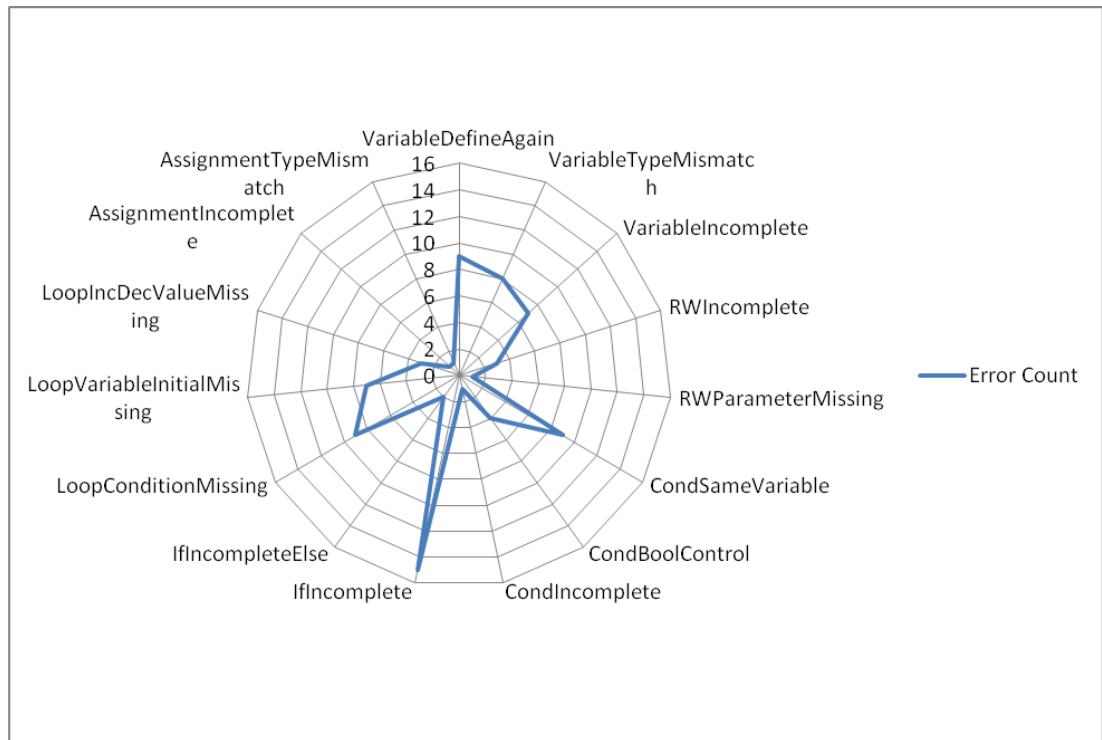


Figure 6.3 Error distribution based on the subfields for Study A

For Study B group, first element of the error type distribution was changed associated with the errors separately depending on the subfields sighted in Table 6.5. Incomplete variable block is highest rate of this distribution. Defining same variable error followed the highest one that is the same order with study A.

Table 6.5 Subfield error rates of Study B

ErrorType	Operation	Rates
1	VariableIncomplete	41,4
4	VariableDefineAgain	20,7
11	CondSameVariable	13,8
5	RWIncomplete	6,9
14	IfIncomplete	3,4
17	LoopVariableInitialMissing	3,4
15	IfIncompleteElse	3,4
8	CondIncomplete	3,4
22	AssignmentIncomplete	3,4

For study B, intensity of the error type is in Figure 6.5. Variable errors are differed from the other operation errors.

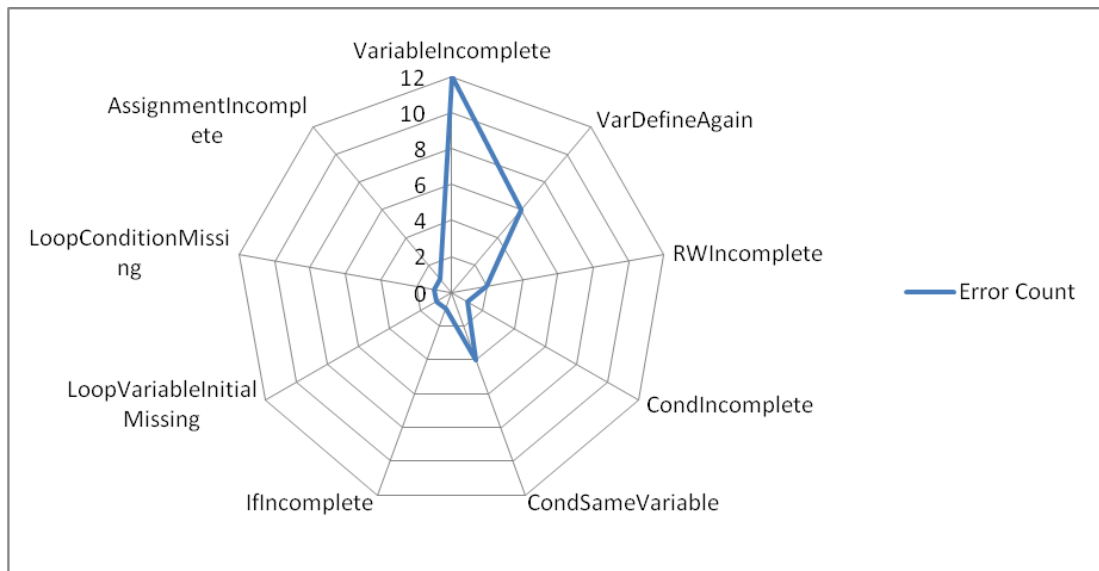


Figure 6.4 Error distribution based on the subfields for Study B

When we examine the errors made by students on generating their algorithm, main topic errors separated from the errors depending on subfields of these main topics. Consequently, the errors, which come from subfields topics, must be considered beside the base concepts rates. In this way child elements, which can be re-worked in an extra lecture with base concepts, can be specified associated with the students activity rates.

6.4 Individual Evaluation & Distribution of Multiple Errors

On comprehension phase, having the knowledge of details of errors is essential for instructors. Beside the errors depending parts as base or subfield, multiple errors made by same student may be important on evaluation. In this concept multiple errors data obtained and evaluated. Firstly, errors examined according to general title groups mentioned previous parts. In second phase, errors analyzed individually for

each students. To examine the multiple errors made by one student in details; firstly duo, trio errors made together determined and an Apriori algorithm, which is a classic algorithm for learning association rules, is used in this study. Shortly to find the relations of errors clearly at this step, data mining is applied.

6.4.1 Apriori Algorithm Result

Firstly, errors examined according to general title groups mentioned in previous parts. In second phase, errors analyzed individually for each students. To examine the multiple errors made by one student in details; firstly duo, trio errors made together determined and an Apriori algorithm, which is a classic algorithm for learning association rules, is used. Shortly to find the relations of errors clearly at this step, data mining is applied. Duo, trio and other combinations of errors for study A group's students have been identified and values in Table6.6 were obtained.

Table 6.6 Relations of multi errors and rates for Study A

Error ID	Rates	Error Type(Base topic)
8 13 17 18 11	2,7	Condition Condition Loop Loop Condition
8 13 17 18	2,7	Condition Condition Loop Loop
8 13 17 11	2,7	Condition Condition Loop Condition
8 13 18 11	2,7	Condition Condition Loop Condition
8 17 18 11	2,7	Condition Loop Loop Condition
13 17 18 11	2,7	Condition Loop Loop Condition
13 17 18	2,7	Condition Loop Loop
13 17 11	2,7	Condition Loop Condition
13 18 11	2,7	Condition Loop Condition
13 11 14	2,7	Condition Condition IfElse
8 13 17	2,7	Condition Condition Loop
8 13 18	2,7	Condition Condition Loop
8 13 11	2,7	Condition Condition Condition
8 17 18	2,7	Condition Loop Loop
8 17 11	2,7	Condition Loop Condition
8 18 11	2,7	Condition Loop Condition
19 18 11	2,7	Loop Loop Condition
17 18 11	2,7	Loop Loop Condition
23 11 14	2,7	Assignment condition IfElse
1 18 11	2,7	Variable Loop Condition

Table 6.6 Relations of multi errors and rates for Study A (cont.)

7 1 14	2,7	ReadWrite Variable IfElse
4 2 14	2,7	Variable Variable IfElse
2 11 14	2,7	Variable Condition IfElse
18 11	8,1	Loop Condition
11 14	8,1	Condition IfElse
13 11	5,4	Condition Condition
2 14	5,4	Variable IfElse
15 14	2,7	IfElse IfElse
23 11	2,7	Assignment condition
23 14	2,7	Assignment IfElse
7 1	2,7	RW Variable
7 14	2,7	RW IfElse
8 13	2,7	Condition Condition
8 17	2,7	Condition Loop
8 18	2,7	Condition Loop
8 11	2,7	Condition Condition
19 17	2,7	Loop Loop
19 18	2,7	Loop Loop
19 11	2,7	Loop Condition
4 2	2,7	Variable Variable
4 14	2,7	Variable IfElse
1 2	2,7	Variable Variable
1 18	2,7	Variable Loop
1 11	2,7	Variable Condition
1 14	2,7	Variable IfElse
13 17	2,7	Condition Loop
13 2	2,7	Condition Variable
13 18	2,7	Condition Loop
13 14	2,7	Condition IfElse
17 2	2,7	Loop Variable
17 18	2,7	Loop Loop
17 11	2,7	Loop Condition
2 11	2,7	Variable Condition

During examining multiple errors, one of the observations is that any student made maximum five errors together. At the same time, duo errors are high density in multiple errors. The highest pair error value is 8.1 and these are belongs to loop-condition and condition-if/else errors. We obtained the inference that if students make an error about the loop or if-else statement, there is an error also in condition

part. Duo errors, which have the second highest value, were condition-condition and variable-if/else errors. We observed that student who has an error on creating condition, repeated this situation when solving a different problem. This indicates that there is miscomprehension for students about condition topics. Another error status was that student who made an error on creating variable, also could not complete if/else part. Depending on the general analysis of multiple errors, condition errors are the highest value. This evaluation also was attained from the multiple errors rates depending on the main topics in Figure 6.5. Numbers on vertical coordinate show the error types and error 3, condition has the highest value.

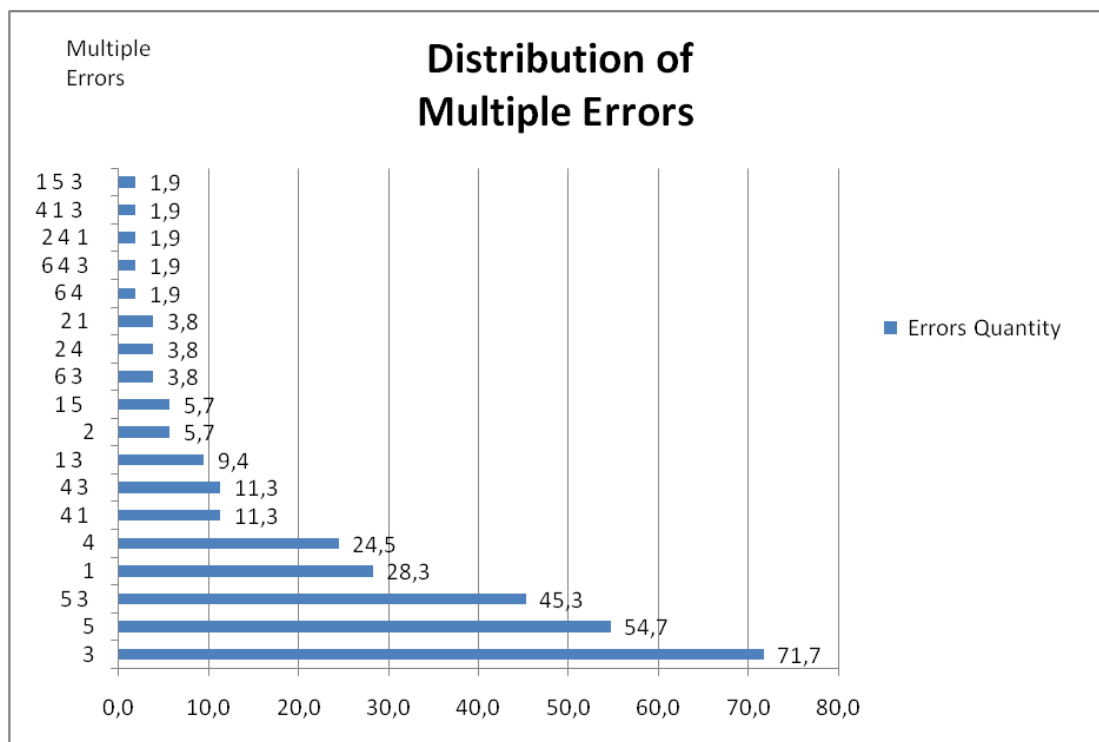


Figure 6.5 Multiple error distribution for Study A

When we examine the errors made by students in study B group, Table 6.7 is obtained for multiple errors. Variable error's rate is the highest value. Variable's operation errors is the highest rates based on the evaluation of main operation's error in previous headings. This is the same in multiple error evaluation.

Table 6.7 Relations of multi errors and rates for Study B

Error ID	Rates	Error Type(Base topic)
22 17 11	6,7	Assignment Loop Condition
8 5 1	6,7	Condition ReadWrite Variable
22 17	6,7	Assignment Loop
22 11	6,7	Assignment Condition
8 5	6,7	Condition RW
8 1	6,7	Condition Variable
5 1	6,7	RW Variable
17 11	6,7	Loop Condition
17 4	6,7	Loop Variable
11 1	6,7	Condition Variable
4 1	13,3	Variable Variable

6.5 Students Evaluation of Algolyzer

A questionnaire is prepared to obtain the feedbacks from the students after they use Algolyzer and this model. This questionnaire includes ten questions related to beneficial rates on learning process of algorithm and programming and essential features of Algolyzer. Students were asked to evaluate the support level of Algolyzer from no benefit to very beneficial shown in Figure 6.6.

1. Did you use Algolyzer educational tool?

Yes
 No

2. How was Algolyzer beneficial for you?

1
 2
 3
 4
 5*

3. Did you have a chance for more practice?

Yes
 No

4. How beneficial were the error messages?

1
 2
 3
 4
 5

5. How do you think dividing problems in operations was beneficial?

1 2 3 4 5

6. How beneficial was writing the programming code independent of PL?

1 2 3 4 5

7. How do you think Algolyzer helped you to develop problem solving skills?

1 2 3 4 5

8. How do you think Algolyzer helped you to improve abstract thinking?

1 2 3 4 5

9. Please check appropriate box(es) which are the most important features of Algolyzer for you?

Generate programming source code independent of PL

Give error messages

Have problem repository

Focus on single problem

Divide solution in operations

10. Please add any other feedback on Algolyzer.

* 1 Almost no benefit 5 Very beneficial

Figure 6.6 Questionnaire of Algolyzer usage

Even though 40 students participated in the questionnaire, only 24 students completed the questionnaire forms. All the students reported that they had a chance for more practice with Algolyzer shown in Figure 6.7.



Figure 6.7 Students feedback on doing more practice

Feedbacks on effectiveness of error messages are illustrated in Figure 6.8. According to students, error messages support them in average level.

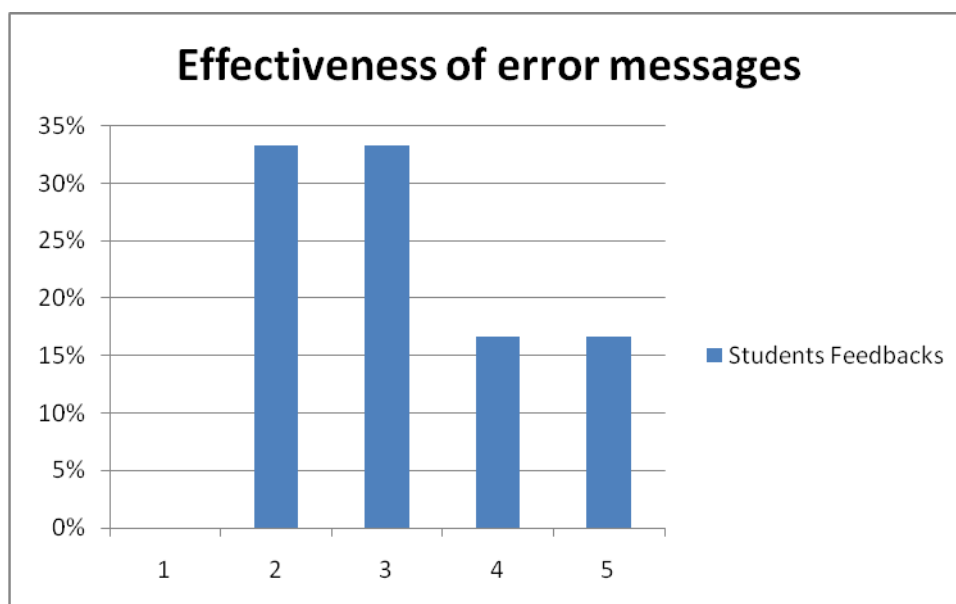


Figure 6.8 Students feedback on effectiveness of error messages

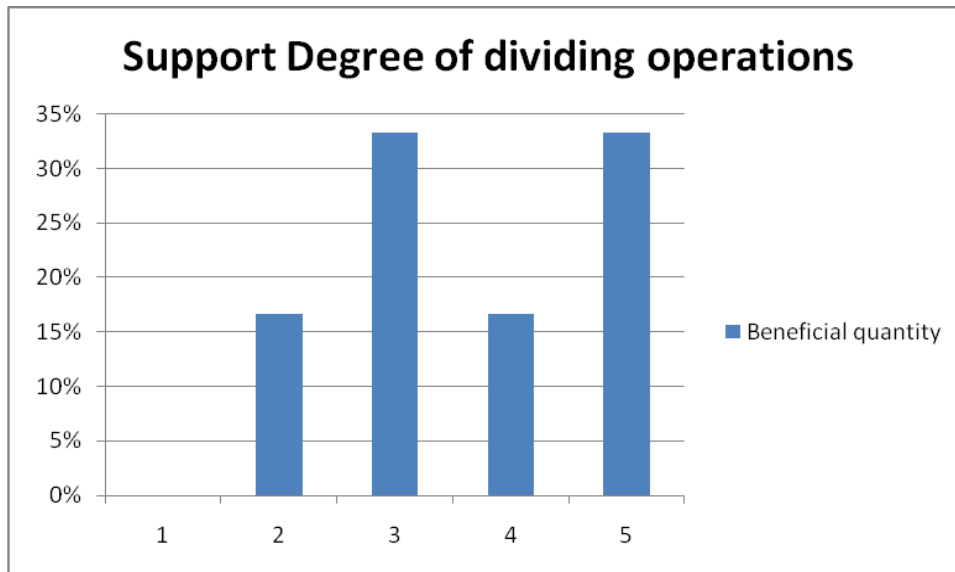


Figure 6.9 Students feedback on support degree of dividing operations

Dividing operations which is the another main feature of the developed tool has been identified as beneficial for participants shown in Figure 6.9. Independent of programming language feature has the same result with dividing operations in Figure 6.10. Students register that writing syntax free code supports them for learning programming process.

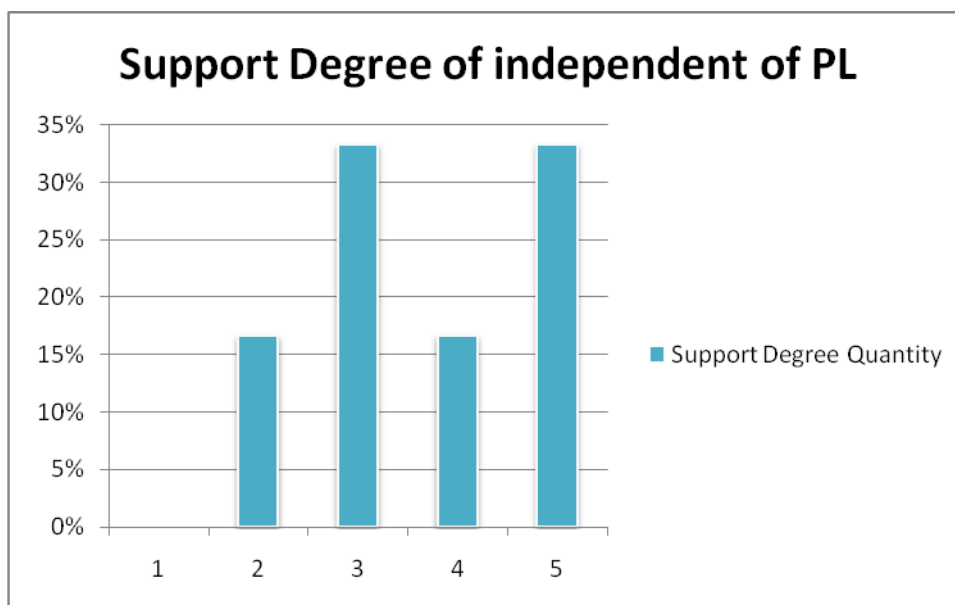


Figure 6.10 Students feedback on support degree of independent of PL

The last rating question's result, which aims to get students opinion on the most important features of Algolyzer is shown in figure 6.11. Giving error message, dividing solution in operations and having problem repository features are the highest and the same percentage.

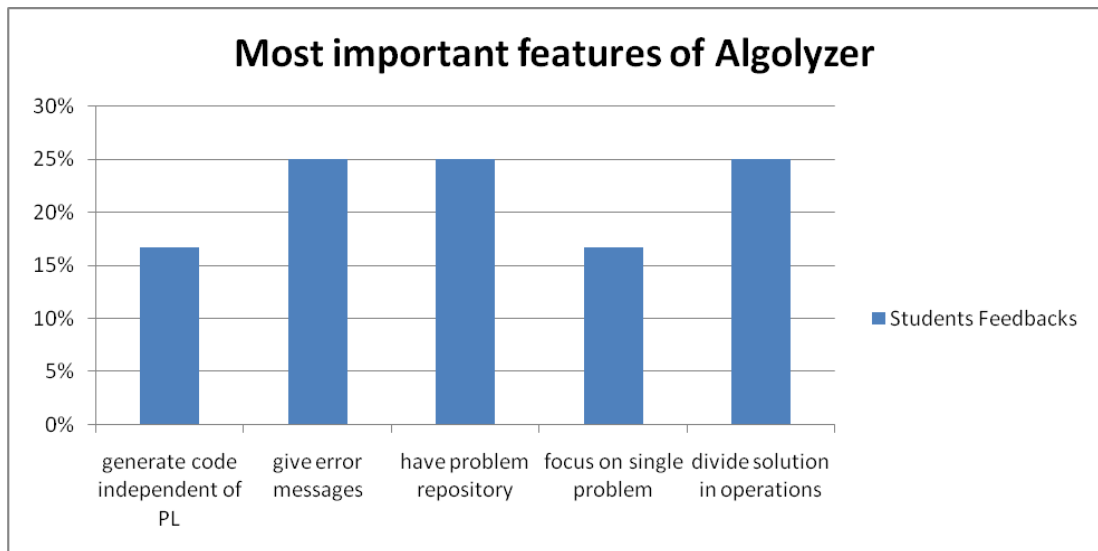


Figure 6.11 Students feedback on important feature of Algolyzer

Some participants responded with their own opinions by answering the question number 10 of the questionnaire; “You may add more new programming language support”, “Dividing operations is excellent”, “This is perfect for starting level students”...

6.6 General Evaluation

Making errors means that there is a misconception or a problem in the learning process. From this point, analyzing the errors of students in details becomes important. As far as obtained result can be summarized as;

- Subfield errors are diversity from main topic errors,
- Multiple errors give information about individual evaluation of students
- Some topics can be depended on the other topic.

The above all general results shared in details with the instructors. The instructors on teaching process can consider these evaluations. In this way, misunderstanding parts can be re-worked in more lectures. At the same time instructor have a chance to form their courses related to miscomprehension parts on the next semesters.

Another important evaluation depends on the survey results. The surveys related to the usage of Algolyzer and its model and the feedbacks prove that Algolyzer supports the students on learning process. The survey results of participants can be summarized as;

- Algolyzer was beneficial for students
- Feature set of the tool is important on learning process
- they had a chance to do more practice which is necessary for improving programming skills

CHAPTER SEVEN

CONCLUSIONS

Teaching and learning programming has never been an obvious process because of the programming is a skill requires practicing and wide effort. Learning design and analysis of algorithms is important for students. Students in computer science disciplines often face difficulties on the related courses. To date, various studies on learning algorithm, programming and data structures have been realized.

In this study, a new method proposed with considering the most important and ineffective parts of the algorithm and programming learning process. Constructivism, which is a learner-centered pedagogy, is used in this study. In this concept, user-friendly, visual interface tool has been developed to support learning process of students. Developed tool, named Algolyzer, provides a programming-language-independent environment for students to create an algorithm solution for predefined algorithmic problem. With this feature, students get a chance to focus on the possible solutions more without facing with programming language syntax issues. This feature helps them in several points as; they understand the importance of main logic of creating an algorithm and decrease time spent on the syntax of the programming language. Another point is that Algolyzer affects the student's motivation and confidence in writing correct programs in a positive way. Thus, students can improve their algorithmic thinking abilities independent from the programming languages.

In the same time, students can work on program codes in different programming languages using multiple language support of Algolyzer. This feature of tool gives a chance to students to do more practice on improving algorithm and programming design skill. By using Algolyzer, students can create algorithmic steps of their solution using visual interface operations. Students can comprehend topics better with separated operations associated with basic concepts of programming.

In addition to this, Algolyzer is also an assistant for instructors. While teaching algorithms and programming, instructors can use this developed tool. Algolyzer saves errors of students occurred during solving the given problems. With having all activities log of each students, instructors can have detailed information on how students use it, where they need more help, what are the lacking parts in the teaching process. In this way, misunderstanding topics can be determined and re-worked in more lectures.

The students of Dokuz Eylül University Computer Engineering and Computer Programming Department have used Algolyzer. At the end of the semester, their usage data has been evaluated. General statistics obtained from the activities and errors of students. The highest error topic was determined. In detail investigation, multiple errors were examined. Selected data mining method applied on collected data and the base problems that happen during designing an algorithm phase were defined. When examining multiple errors we observed that pair's errors are high density in multiple errors. Another result is that, highest error type in evaluation on multiple errors is different from highest one in evaluation based on general error type definition.

Another point of this study is that, in verbal form we attained the feedback from the students that they used the tool and benefited from the problem repository for preparing the algorithm and programming exams.

In summary, in this thesis during the study a tool has been developed on learning algorithms and programming area to support students to learn basics of programming and to assist instructors during their teaching classes. The developed tool has been developed as a web based application and can be used in both distance learning courses and in-class sessions. This developed tool provides instant repeat chance to students if used in distance learning sessions.

The most significant contributions of this study can be summarized as

- Developed tool can bridge the gap between the learning process and learning complications.

- Attain knowledge of errors of students related with comprehension while developing a program
- Give a chance to enhance success rate with repeating misunderstanding parts and increasing student's motivation.

Some surveys related to usage of Algolyzer and its model has been applied to students. The feedbacks of students prove that Algolyzer supports the students during algorithm and programming learning process effectively.

In the future, adding new features will help catching more logic errors to be used for the study and will result with better-analyzed problems on improving algorithmic thinking phase. Collecting more data will always be in our plans to create results that are more sensible and help to propose new models. Another primary goal in this study will be improving real time notifications and improving user interactions with enabling more validations rules. Adding more programming languages is another mid-term plan. We plan to improve Algolyzer to make it a standard tool in starter-level algorithms and programming courses. Drawing flowchart feature, which is in desktop-based version of this study, will be added to the web-based version. Every student - does not matter which discipline they are in - face different problems during the learning process. Students on various disciplines at engineering can use the developed tool and depending on their usage and logged errors; new inferences will be obtained for next studies.

REFERENCES

- Al-Imamy, S., Alizadeh, J., & Nour, M. A. (2006). On the development of a programming teaching tool: The effect of teaching by templates on the learning process. *Journal of Information Technology Education, Vol. 5*, p. 271-283.
- Aispuro, E. E., Licea, G. , Sus´rez, J., Sandoval, A., Carren˜o, M. A., & Estrada, I. (2009). Supporting the development of interactive applications in introductory programming courses. *Computer Applications in Engineering Education, 20*, p.214-220.
- Ben-Ari, M. (1998). Constructivism in computer science education. *Proceedings of the twenty-ninth SIGCSE technical symposium on computer science education*, p. 257-261.
- Biggs, J. B. (2003). *Teaching for quality learning at university. what the student does*. Maidenhead, United Kingdom: Open University Press, p. 13.
- Bloom, B. S. (1956). *Taxonomy of educational onjectives, handbook1: The Cognitive Domain*. White Plains, N.Y: Longman.
- Byrne, M. D, Catrambone, R., & Staskoc, J. T. (1999). Evaluating animations as student aids in learning computer algorithms. *Computers & Education, Vol. 33*, p. 253-278.
- Chou, C. Y., & Sun, P. F. (2010). An educational tool for visualizing students’ program tracing processes. *Computer Applications in Engineering Education, 2010*, doi:10.1002/cae.20488.
- Chou, P. H. (2002). Algorithm education in Python. *Proceedings of Python 10*, p. 177-185.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms (3rd ed.)*. Cambridge: The MIT Press.

- Costa, C. J., Aparicio, M. & Cordeiro C. (2012). Web-based graphic environment to support programming in the beginning learning process. *Proceedings of the 11th international conference on Entertainment Computing, LNCS Vol. 7522*, p. 413–416.
- Dagdilelis, V., Satratzemi, M., & Evangelidis, G. (2004). Introducing secondary education students to algorithms and programming. *Education and Information Technologies*, 9:2, 159-173.
- Elgamal, A. F., & Abas, H. A., & Baladogh, E.. M. (2013). An interactive e-learning system for improving web programming skills. *Educational Information Techonology*, 18, p. 29-46.
- Ferna'ndez, A. J., & Sa'nchez, J. M. (2004). CGRAPHIC: Educational software for learning the foundations of programming. *Computer Applications in Engineering Education*, 11, p. 167-178.
- Gomez-Albarra'n, M. (2005). The teaching and learning of programming: A survey of supporting software tools. *The Computer Journal*, Vol. 48, p. 130-144.
- Holvikivi, J. (2010). Conditions for successful learning of programming skills. *Key Competencies in the Knowledge Society*, Vol. 324 Springer, 155-164.
- Hundhausen, C. D., & Brown, J. L. (2008). Designing, visualizing, and discussing algorithms within a CS 1 studio experience: An empirical study. *Computers & Education*, 50, p. 301-326.
- Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002) .A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13, p.259-290.
- Hübscher-Younger, T., & Narayanan, N. H. (2003). Constructive and collaborative learning of algorithms. *ACM Special Interest Group on Computer Science Education (SIGCSE'03)*, p. 6-10.

- Jain, A. K., Singhal, M., & Gupta, M. S. (2010). Educational tool for understanding algorithm building and learning programming languages. *World Congress on Engineering and Computer Science, Vol. I*, 292-295.
- Jenkins, T., (1998). A participative approach to teaching programming. *Proceedings of the 6th Annual Conference on the Teaching of Computing and the 3rd Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITICSE'98)*, p. 125-129.
- Jerez J. M., Bueno D., Molina I., Urda, D. & Franco, L. (2012). Improving motivation in learning programming skills for engineering students. *International Journal of Engineering Education, Vol.28, No.1*, p. 202-208.
- Jonassen, D. H. (1996). *Computers in the classroom: Mind tools for critical thinking*. Columbus, OH: Merrill/Prentice-Hall.
- Kordaki, M. (2010). A drawing and multi-representational computer environment for beginners' learning of programming using C: Design and pilot formative evaluation. *Computers & Education, 54*, p.69-87.
- Kordaki M., Miatidis, M., & Kapsampelis, G., (2008). A computer environment for beginners' learning of sorting algorithms: Design and pilot evaluation. *Computers & Education, 51*, p.708-723.
- Lahtinen, E., AlaMutka K., & Järvinen, H. (2005). A study of the difficulties of novice programmers. *Proceedings of the 10th Annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE 05), Portugal*, p. 14-18.
- Lazaridis, V. , Samaras, N., & Sifaleras, A. (2010). An empirical study on factors influencing the effectiveness of algorithm visualization. *Computer Applications in Engineering Education, DOI 10.1002/cae.20485*.

- Li, J., & Zhang, Z. (2010). A learning tool of genetic algorithm. *Second International Workshop on Education Technology and Computer Science (ETCS), Vol.1*, p. 443-446.
- Li, J., & Liu, W. (2009). An educational tool for the ant colony optimization algorithm. *First International Workshop on Education Technology and Computer Science (ETCS), Vol.1*, p. 55-58.
- Licea, G., Juárez, J. R., Martí'nez, L. G., & Aguilar, L. (2008). Developing programming tools to reach a deeper understanding of advanced programming concepts. *Computer Applications in Engineering Education, 16*, p. 305-314.
- Licea, G., Juárez-Ramírez, R., Gaxiola, C., Aguilar, L., & Martí'nez L. G. (2011). Teaching object-oriented programming with AEIOU. *Computer Applications in Engineering Education*, doi:10.1002/cae.20556.
- Marcelino, M., Gomes, A., Dimitrov, N. & Mendes, A., (2004), Using a computer based interactive system for the development of basic algorithmic and programming skills. *Proceedings of International Conference on Computer Systems and Technologies (CompSysTech'2004)*, p. 1-6.
- Milne, I., & Rowe G., (2002). Difficulties in learning and teaching programming - Views of students and tutors. *Education and Information Technologies, Vol.7*, p. 55-66.
- Müldner, T., Shakshuki, E., & Kerren, A. (2008). Algorithm education using structured hypermedia. *Advances in Distance Education Technologies Series, Chap. 5, IGI Global*, 58-84.
- Nardi, B. A. (1996). Studying context: A comparison of activity theory, situated action models, and distributed cognition. In B. A. Nardi (Ed.), *Context and consciousness: Activity theory and human-computer interaction*. Cambridge, MA: MIT Press.

- Ng, S.C., Choy, S.O., Kwan, R., & Chan, S.F. (2005). A web-based environment to improve teaching and learning of computer programming in distance education. *International Conference on Web-based Learning, Vol. 3583 Springer*, p. 279-290.
- Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers*. Dordrecht: Kluwer Academic Publishers.
- Perrenet, J., Groote, J. F., & Kaasenbrood, E. (2005). Exploring students' understanding of the concept of algorithm: Levels of abstraction. *10th Annual Conference on Innovation and Technology in Computer Science Education*, p. 64-68.
- Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, Vol. 13, No.2*, p.137-172.
- Sajaniemi, J., & Hu, C. (2006). Teaching programming: Going beyond "objects first". *18th Workshop of the Psychology of Programming Interest Group, University of Sussex*, p. 255-265.
- Satratzemi, M., Dagdilelis, V., & Evagelidis, G. (2001). A system for program visualization and problem-solving path assessment of novice programmers. *6th Annual Conference on Innovation and Technology in Computer Science Education, ACM*, 137-140.
- Satratzemi, M., Xinogalos, S., & Dagdilelis, V. (2003). An environment for teaching object-oriented programming: ObjectKarel. *The 3rd IEEE International Conference on Advanced Learning Technologies*, p. 342-343.
- Selby, C., C., (2011). Four approaches to teaching programming. *Learning, Media and Technology: A doctoral research conference, London*.

- Shabanah, S., & Chen, J. X. (2009). Simplifying algorithm learning using serious games. *Proceedings of the 14th Western Canadian Conference on Computing Education, ACM*, p. 34-41.
- Shakshuki, E., Kerren, A., & Müldner, T. (2007). Web-based structured hypermedia algorithm explanation system. *International Journal of Web Information Systems, Vol.3, No.3*, p 179-197.
- Sutinen, E., Tarhio, J., & Terasvirta, T. (2003). Easy algorithm animation on the web. *Multimedia Tools and Applications, Vol.19*, p. 179-194.
- Taherkhani, A., Korhonen, A., & Malmi, L. (2010). Recognizing algorithms using language constructs, software metrics and roles of variables: An experiment with sorting algorithms. *The Computer Journal, Vol. 54, No.7*, p. 105-1066.
- Truong, N., Bancroft, P., & Roe, P. (2003). A web based environment for learning to program. *ACSC '03 Proceedings of the 26th Australasian computer science conference, Vol. 16*, p. 255-264.
- Van Gorp, M., J., & Grisson, S., (2001). An empirical evaluation of using constructive classroom activities to teach introductory programming. *Computer Science Education, Vol. 11, No. 3*, p. 247-260.
- Vygotsky, L. S. (1978). *Mind and society: The development of higher mental processes*. Cambridge, MA: Harvard University Press.
- Wang, Y., Li H., Feng, Y., Jiang, Y., & Liu, Y. (2012). Assessment of programming language learning based on peer code review model: Implementation and experience report. *Computers & Education, 59*, p. 412-422.
- White, S., Martinez, T., & Rudolph, G. (2012). Automatic algorithm development using new reinforcement programming techniques. *Computational Intelligence, Vol. 28, Issue 2*, p. 176-208.

- Winslow, L. E. (1996). Programming pedagogy – A psychological overview. *SIGCSE Bulletin, Vol. 28 Issue 3*, p.17–22.
- Wulf T. (2005). Constructivist approaches for teaching computer programming. *Proceedings of the 6th conference on Information technology education SIGITE 05*, p.245-248.
- Yıldırım, M., & Kut, A. (2010). An interactive education tool for conventional/distance learning. *The International Symposium on Open and Distance Learning(IODL), Eskişehir*, p. 903-908.
- Ziegler, U., & Crews, T., (1999), An integrated program development tool for teaching and learning how to program. *Proceedings of the 30th SIGCSE Technical Symposium on Computer science education, Vol. 31*, p. 276-280.

APPENDICES

A. List of Abbreviations

EXE	Executable
GUI	Graphical User Interface
IDE	Integrated Development Environment
IO	Input Output
LHS	Left Hand Side
OOP	Object Oriented Programming
PL	Programming Language
RHS	Right Hand Side
RW	Read Write

B. Class Diagram Of Algolyzer

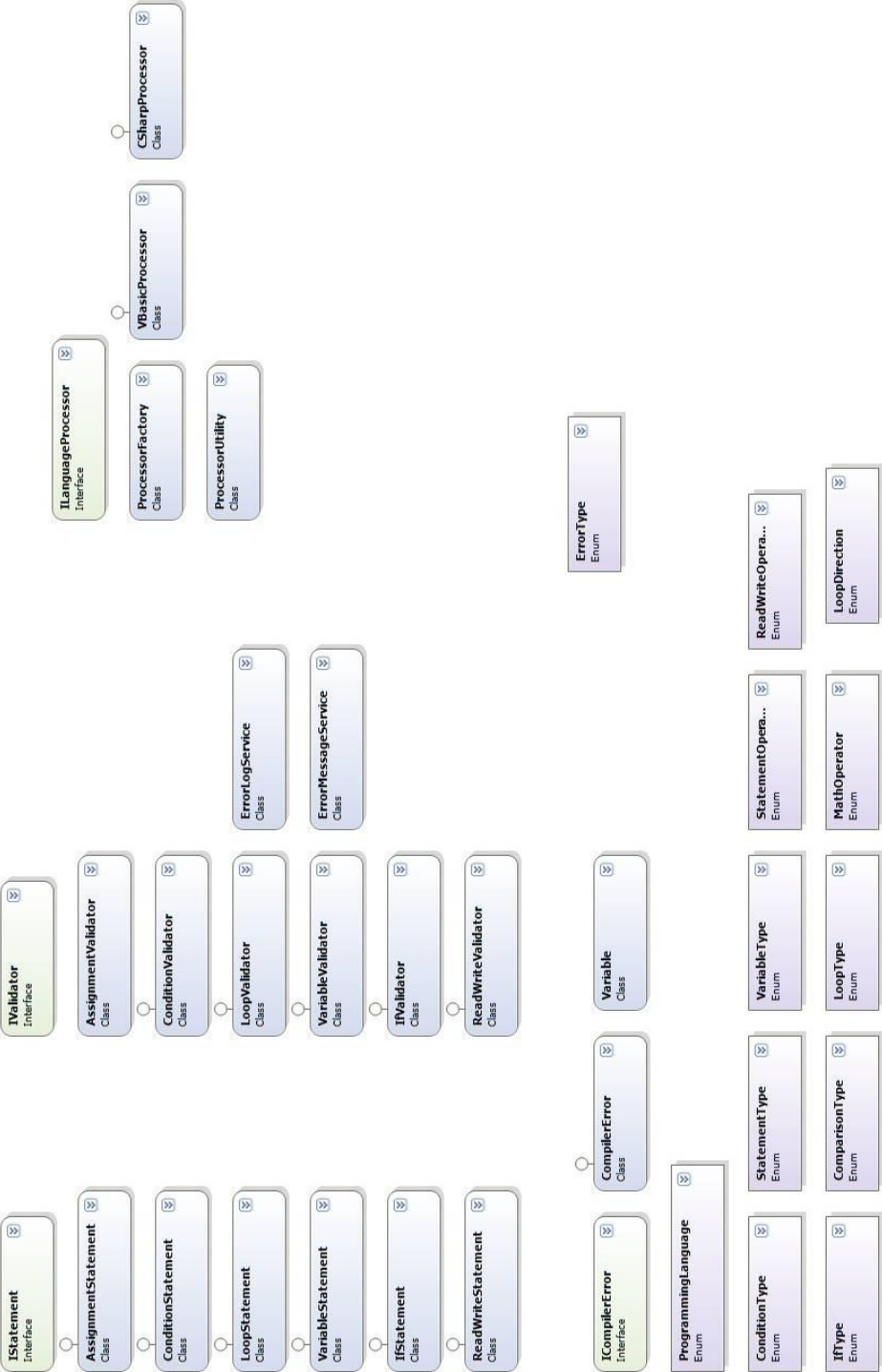


Figure A.1 Class Diagram of Algolyzer includes all classes

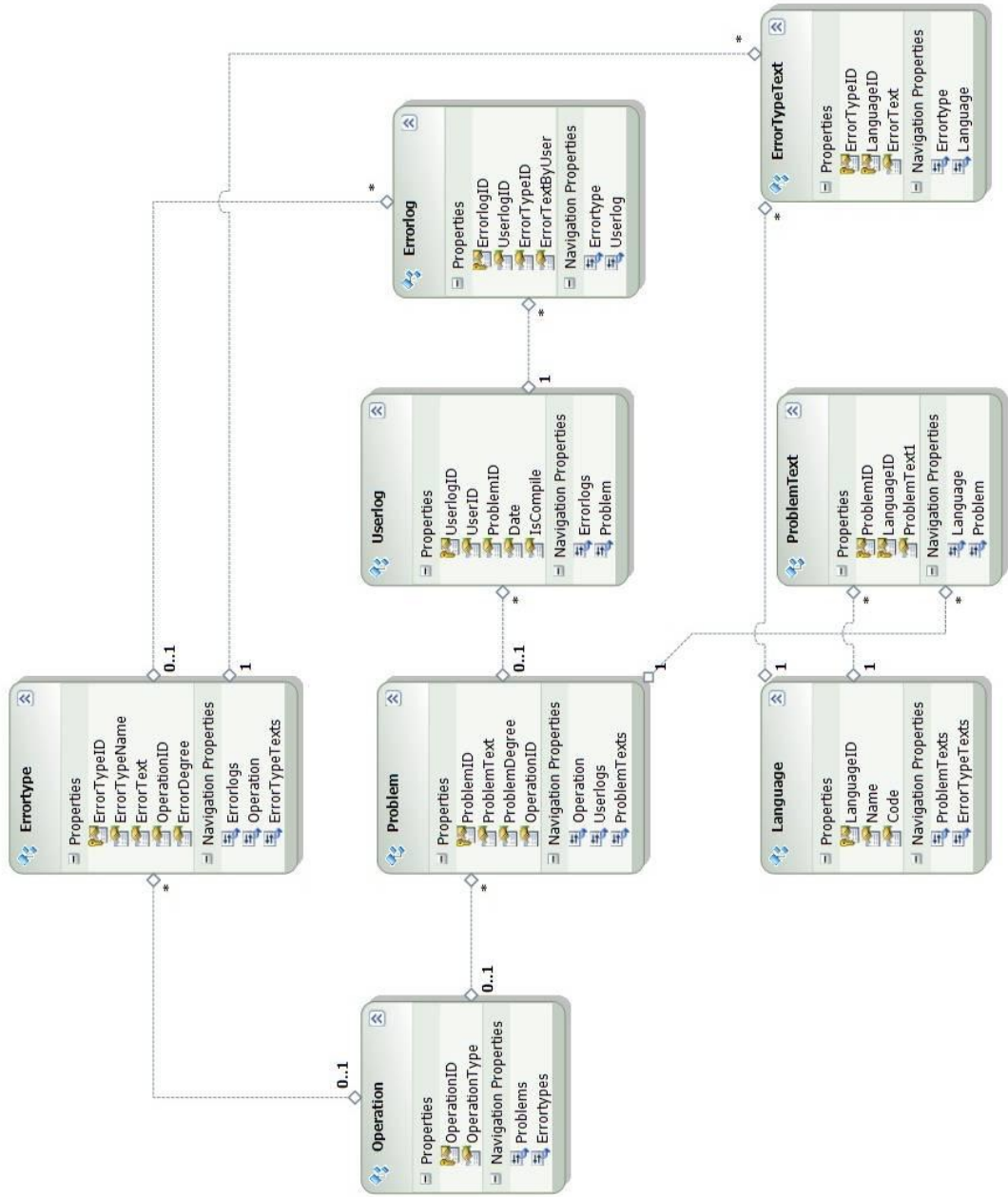


Figure A.2 Class Diagram of Data Module with Properties

C. Algolyzer Usage

In this section, one scenario was taken to show the usage of Algolyzer. Instance screen shots illustrated in the following figures.

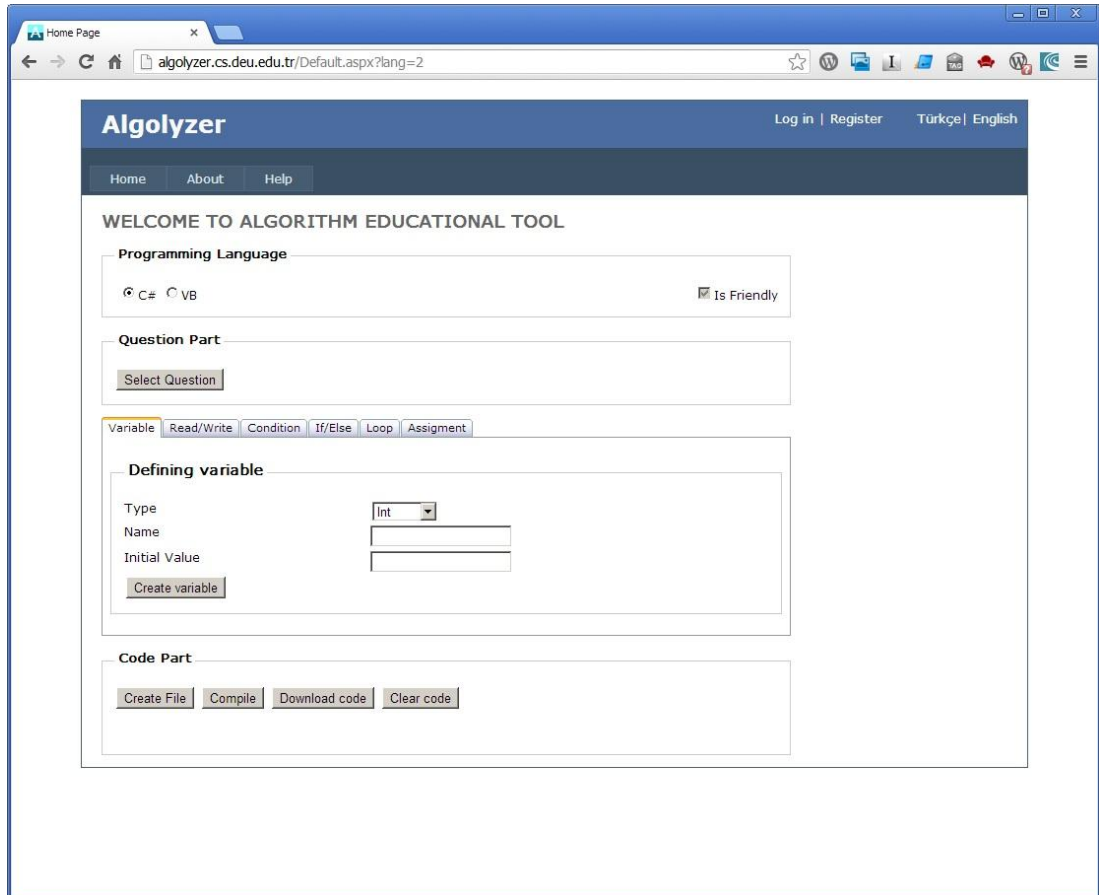


Figure B.1 Start Window of Tool

Students firstly try to understand problem given by the system, and work on problem and generate solution, than student pass the next problem. Extensive question repository can be constituted on algorithm and programming area for students and instructors. At the beginning of the tool' usage, students determine main operations with solution separation task. This method supports the students for improving a solution algorithm easily.

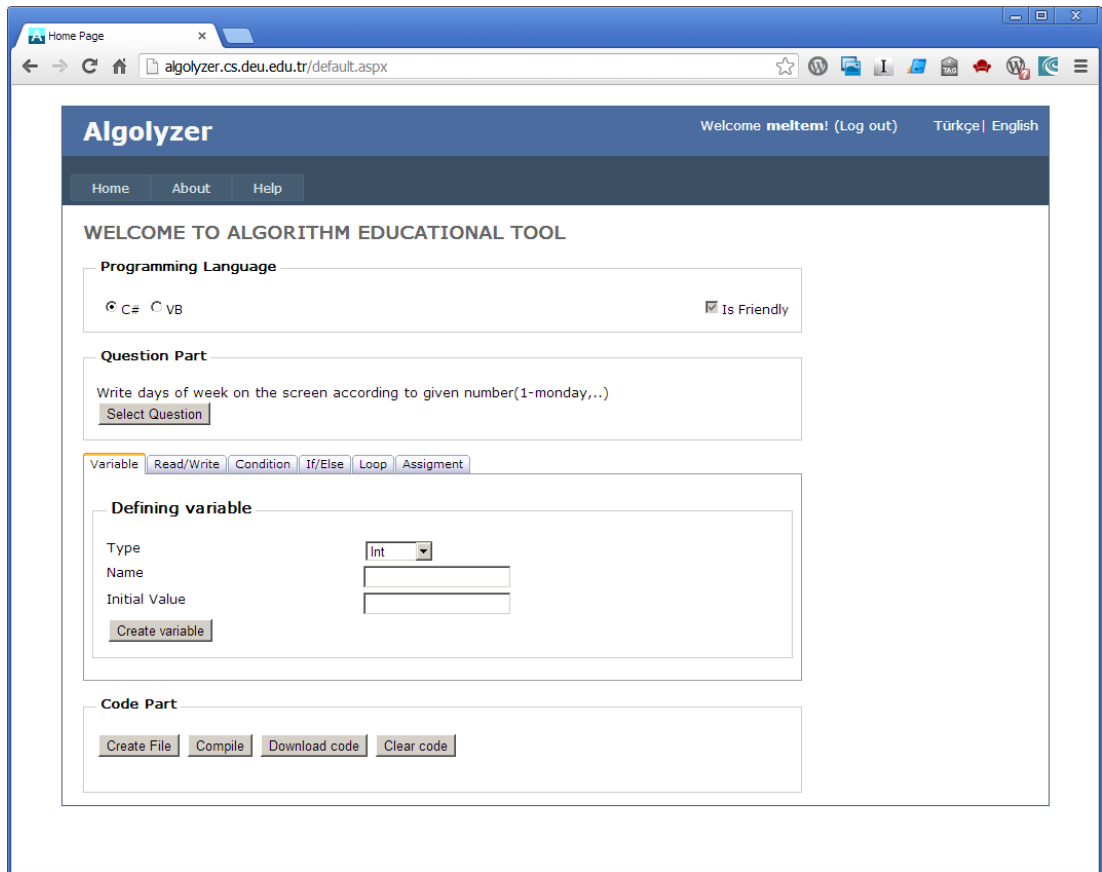


Figure B.2 View of Algolyzer with Selecting Problem after Login the Page

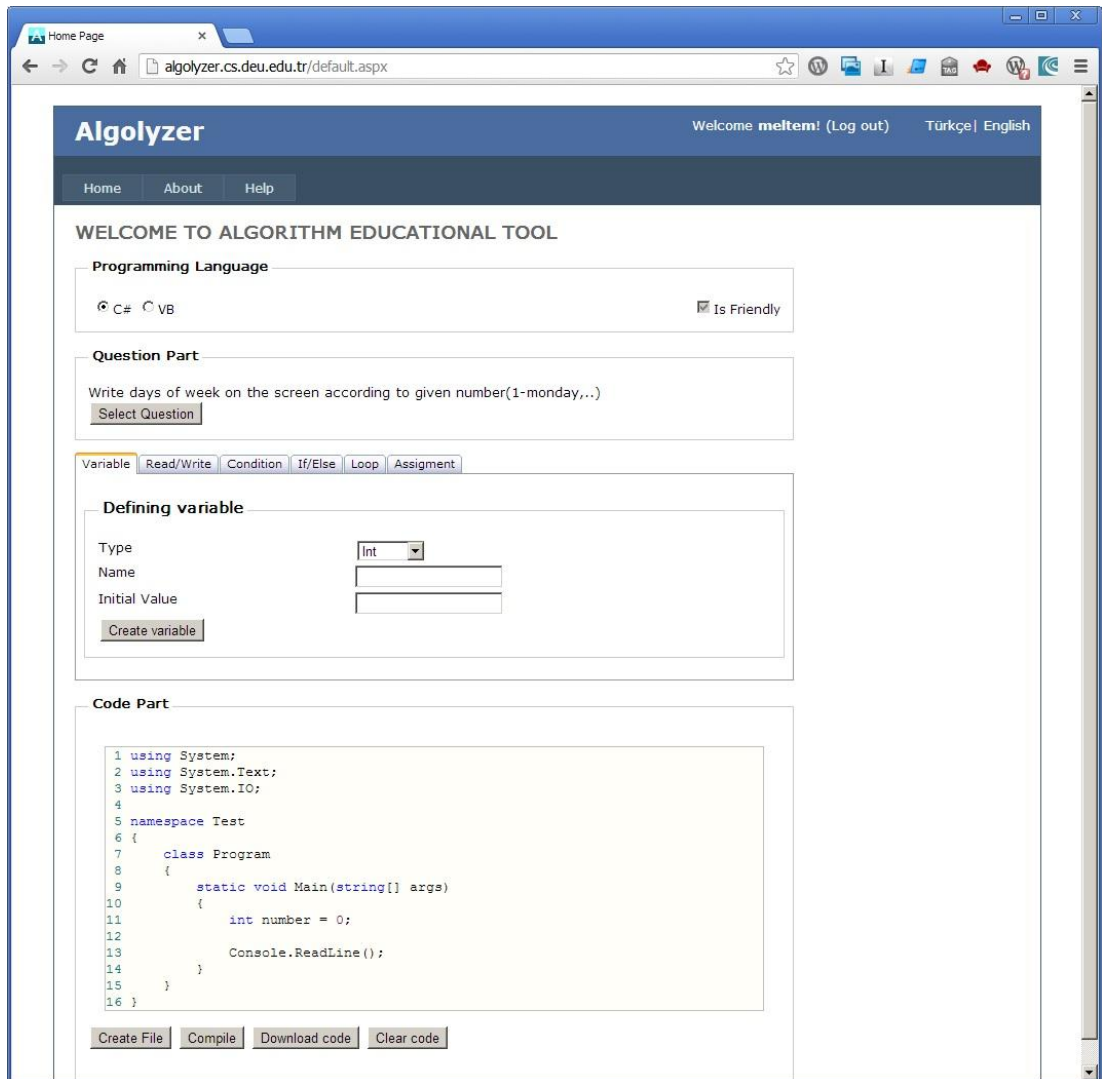


Figure B.3 Beginning the Writing Code with Operations Support View

D. Previous Version of The Study

The first version of this study implemented as a desktop-based application. The general features as problem, operation and code parts improved in this first prototype. Welcome screen can be seen in Figure C.1.

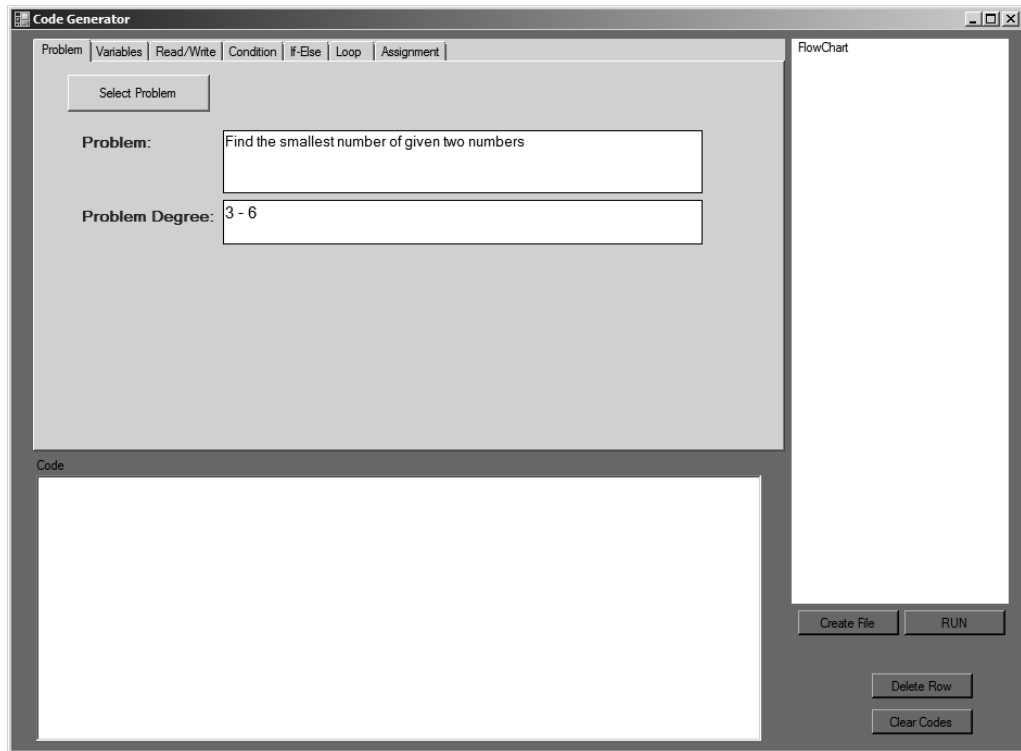


Figure C.1 Welcome screen of the application

Operation features are same with the last version of the application. Read/write statement creation is shown in figure C.2.

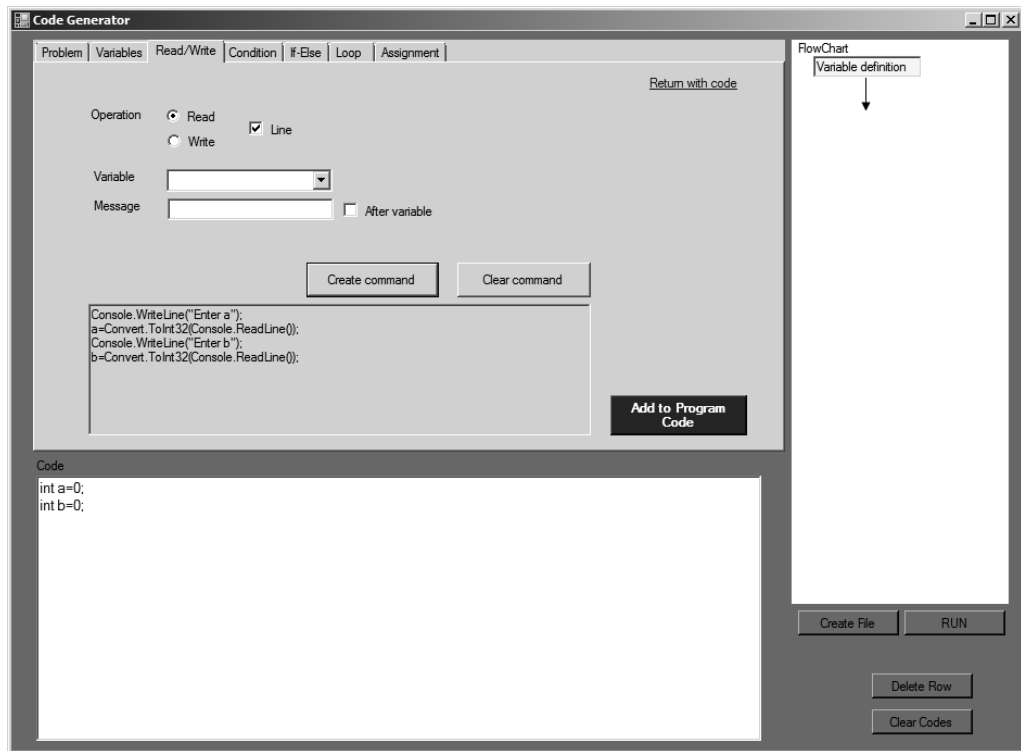


Figure C.2 Sample screen of Read/Write statement creation

There are some different features at this version as flowchart. When creating code part; flowchart of this algorithm was drawing simultaneously on the screen. Thus, each step of algorithm can be followed on flowchart field.

Another sample screen illustrated in Figure C.3. There are three steps for if statement. When creating if code block, system adds related shape in flowchart area automatically.

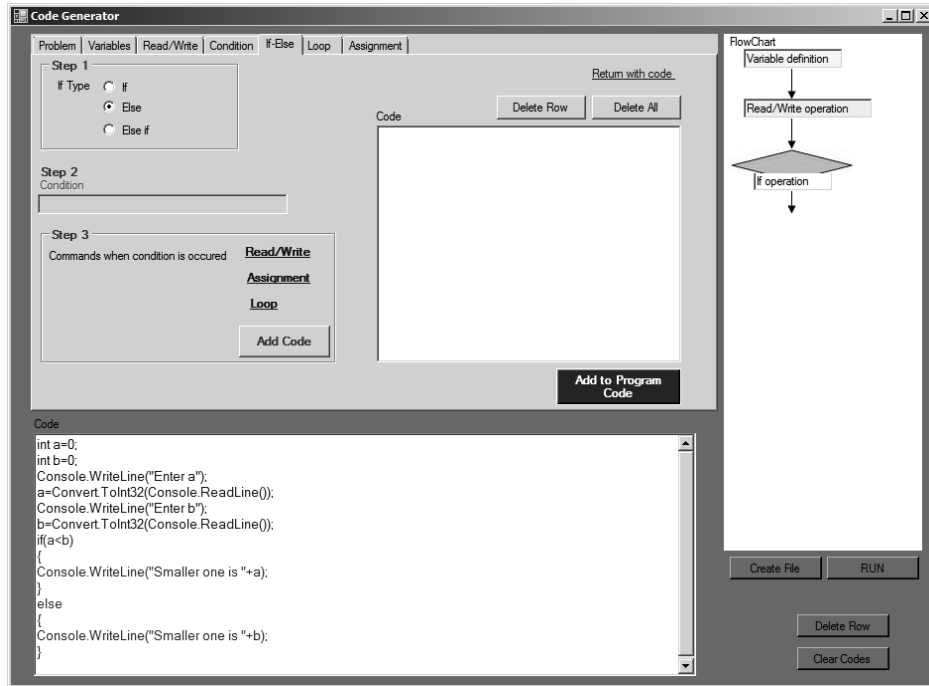


Figure C.3 If statement view with flowchart presentation

Realizing debug operation is another different feature from final version of the application. Students can debug their generated code file.

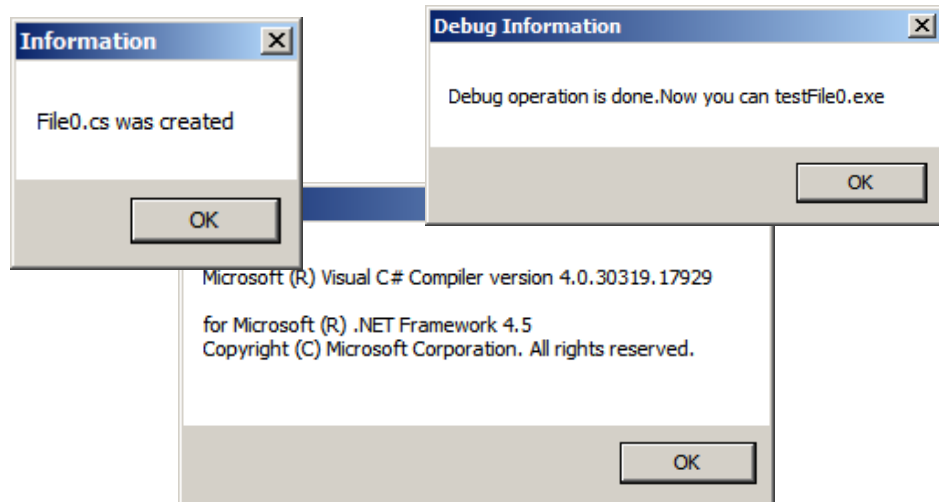


Figure C.4 Create source file and debug operations

After debugging operation, students can run executable file created by the system at user side. Students can obtain executable files from application path. On console students enter input values and take output with running executable file illustrated in Figure C.5 and Figure C.6.

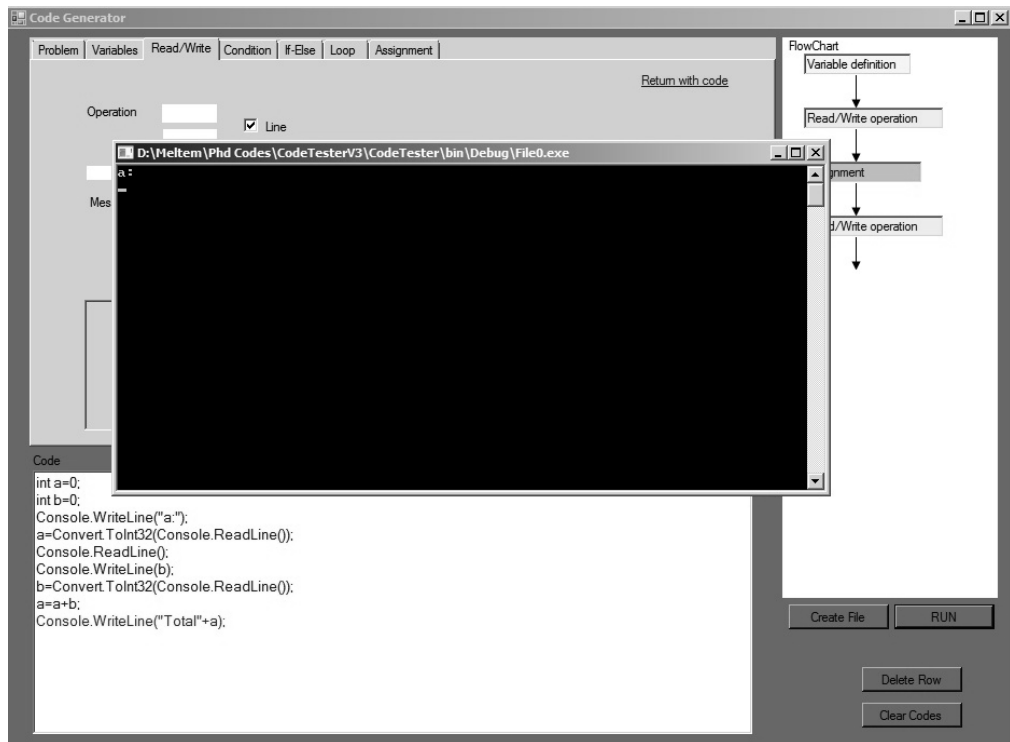


Figure C.5 Run executable file view

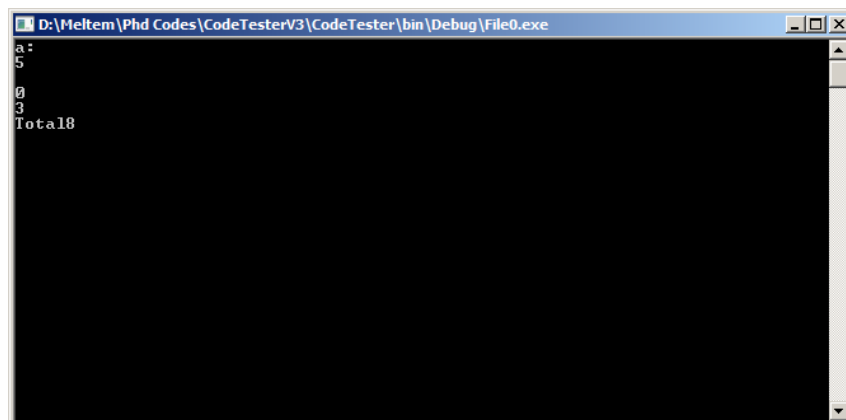


Figure C.6 Realize IO operations on exe file