Journal of Quality and Technology Management
Volume VIII, Issue II, December 2012, Page 135–151

A QUALITATIVE FRAMEWORK FOR INTRODUCING PROGRAMMING LANGUAGE AT HIGH SCHOOL

M.S. Farooq[1], A. Abid[1,3], S.A. Khan[1], M.A. Naeem[4], A. Farooq[1], K. Abid[2] & M. Shafiq[5]
[1]University of Central Punjab, Lahore, Pakistan
[2]University of the Punjab, Lahore, Pakistan
[3]Department of Electronics and Information, Politecnico di Milano, Milan, Italy
[4]Department of Electronics and Electrical Engineering, University of Glasgow, UK
[5]Institute of Quality & Technology Management, University of the Punjab, Lahore, Pakistan

ABSTRACT

Programming is rapidly becoming an integral part of the basic knowledge that should be possessed by the students of all disciplines, in general, and of engineering and sciences, in particular. This is evident from the fact that the programming courses were included as a part of bachelor's degree curriculum and soon after, an introductory course of programming has been introduced at high school level (grade X and XI). In this paper, we highlight the point that in the near future programming should be considered among the fundamental courses in the curriculum of schools from grade VI. This course should be aligned with the basic principles of mathematics, which have already been learned by the students.

As our main contribution, we present a qualitative framework for designing such programming language at school level. We also discuss the possibility as to how an existing programming language should be customized for the better learning of the high school students. In this regard, we have outlined the major requirements including language's theoretical design and implementation, tools related to the language, curriculum design and assessment.

Keywords: *Language design framework, programming at high school, programming language.*

1) INTRODUCTION

Computer programming is increasingly becoming a need for the people belonging to all fields of life. The reason is that ever since the IT revolution almost everything has been digitized and the use of software is a usual activity for the people from all domains, as there are software systems available for a simple small shop to a big high-tech hospital, and similarly from a simple telephone exchange to space navigation systems. Furthermore, most of the modern general purpose software, which are used by a wide variety of users e.g. MS Office involves some basic level programming to enrich documents with different features. Similarly, researchers from different domains medical sciences, neurologists, geologists etc. not only use computer software systems, but they also have to write their own customized software systems through which they manage their own research experiments. Basic understanding and knowledge of computer programming is a desirable for the researchers from all domains. Above all, it is increasingly becoming evident that a basic understanding of computer programming is a need for the people belonging to various different domains of life. Therefore, we argue that the programming language course should be introduced at high school level (Peter Brusilovsky, 1998).

In order to learn computer programming in a better way one should start learning it at high school level (Elkner, 2001). There are some computer languages which are being taught at high school level in Europe and the US (Linda Grandell, 2006) (programming at high school, 2010), whereas some other languages are taught at other levels of education as shown in Table 1.

Table 1: Widely used Programming Languages at Different Levels of Education

Educational Level	**Appropriate Languages**
Pre-School to Grade 2	Logo, Karel, Scratch
Grades 2 to 4	Logo, Scratch, Karel 3D
Grades 5 to 8	Mama, Greenfoot, Learning Oriented BASIC
High School	Squek, Microsoft Small BASIC, Tortoise
College Freshmen, University	Turingal, Karel Genie, Wayfarer

Once we explore the programming language being taught at high school level, we can figure out that some of them are fun based languages with

the main purpose to involve the kids to interact with computer (Xiaoxia Wang, 2011) (Li-Chun Wang, 2010). Whereas, on the other end, some of the programming languages use complex programming syntax which is generally hard to learn and grasp for a student of high school (David Smith, 2010). Programming language involves different kind of understanding and learning as compared to the natural languages. In terms of teaching the first course in computer programming, initially, a student is introduced to the notions of pseudocode and algorithm, which mainly define the workflow or the steps involved in order to solve a problem in easily understandable manner. Then the students are taught how to implement a pseudocode in a particular language. From thereon more features are taught to the students to understand the language in more details (SeungWook Yoo, 2006).

In this paper, we have presented a qualitative framework which holds the key features that a programming language should possess to be used at high school level. Our framework suggests that a language should be simple and clean; should have quick feedback mechanism; should be a scripting language instead of an imperative language; should be a subset of an existing high level programming language; should be aligned with the syllabus of each grade.

The rest of the paper is presented in the following order. The next section discusses the related work and main concerns related to the design of the language; Section 3 presents our main framework for the designing and assessment of ideal programming language for high school students. In Section 4, we discuss the proposed framework. Whereas, Section 5 concludes the paper and provides the future directions that emerge from this work.

2) RELATED WORK AND LANGUAGE DESIGNING ISSUES

Popularity and adoptability of programming language for beginners depends upon the features provided by the language. So we consider that as a main feature in designing an ideal programming language for novice programmers should be easy to understand and it should especially focus on minimizing the common mistake made by the novice programmers. This can be achieved if the proposed language has a simple syntax, simple words and involves simple implementation standards. There should be minimal or no learning over head for learning programming language. It

should be seriously considered that a novice programmer does not have any programming or computer science background (Chen, 2004). Therefore, while designing the programming language, the designers should keep in mind that the pre-requisites for learning the language should be minimized (Schollmeyer, 1996).

Many efforts have been made to develop Mini languages for high school level. Here Mini langue term is used for languages that use some actor (turtle or robot) which performs different operations at micro-world using some predefined commands. The main purpose of such approaches is to help the novice programmers learn programming with ease. (Mendelson P.G., 1990). Logo (the turtle) (Papert, 1980) was the first Mini Language. Originally, Logo was not designed for educational purpose but it provides a good start for other Mini languages. Karel the Robot, introduced by (Pattis, 1981) was originally designed for educational purpose. The basic objective of Karel was to learn Pascal which was very popular language at that time.

There are several mini-languages which were directly stirred by Karel and use some of its features: Martino (Olimpo, 1988) and Marta (Calabrese, 1989) in Italy, Darel (Kay, 1993) in Australia and Karel-3D (Hvorecky, 1992) in Slovakia. However, all these languages have a common drawback that they do not have the concepts of variables and parameter passing. Thus, they fail to impart actual programming concepts to the students (Clancy, 2004).

The Karel Genie has been in used in secondary schools and universities throughout the US for nearly ten years. It has been used in teaching computer programming at several renowned universities including Carnegie Mellon University, Harvard University, Stanford University, New York University, Ohio State, Swarthmore College and some other institutions. Figure 1 shows the variants of Karel language which are used at different educational levels.

In order to design a programming language for the high school students it should be strictly aligned with the course contents of other subjects being taught, especially with mathematics. Therefore, as soon as they learn mathematical constructs they should be able to learn their corresponding constructs in the programming language. Furthermore, in terms of teaching, the concepts should be introduced step by step, e.g.

definition, variables introduction to conditional control statements and loops should be introduced step by step, and based on the fact that the students have learned necessary ingredients from other courses to understand such concepts.

Naturally, a language designed for the students of high school will possess simpler features and more advanced features will be introduced at later stages. This will help in many ways: firstly, the students will have sufficiently long time to grasp a concept and its corresponding implementation in a programming language; secondly, with the passage of time the student becomes more familiar with the programming language as well as programming techniques; thirdly, this provides a smooth run from a novice programmer to a professional programmer. Generally, it has been observed that the students who have not learned computer programming at early stages of their education face more problems at higher levels as well (Mitchel Resnick, 2009).

Apart from the above mentioned objectives, we also intend to design the language in such a way that it should improve the thinking power, logical reasoning and problem solving skills of the students. This is somewhat related to the design of the language and it is more closely related to the teaching methodology and alignment of the syllabus of programming course with other relevant courses.

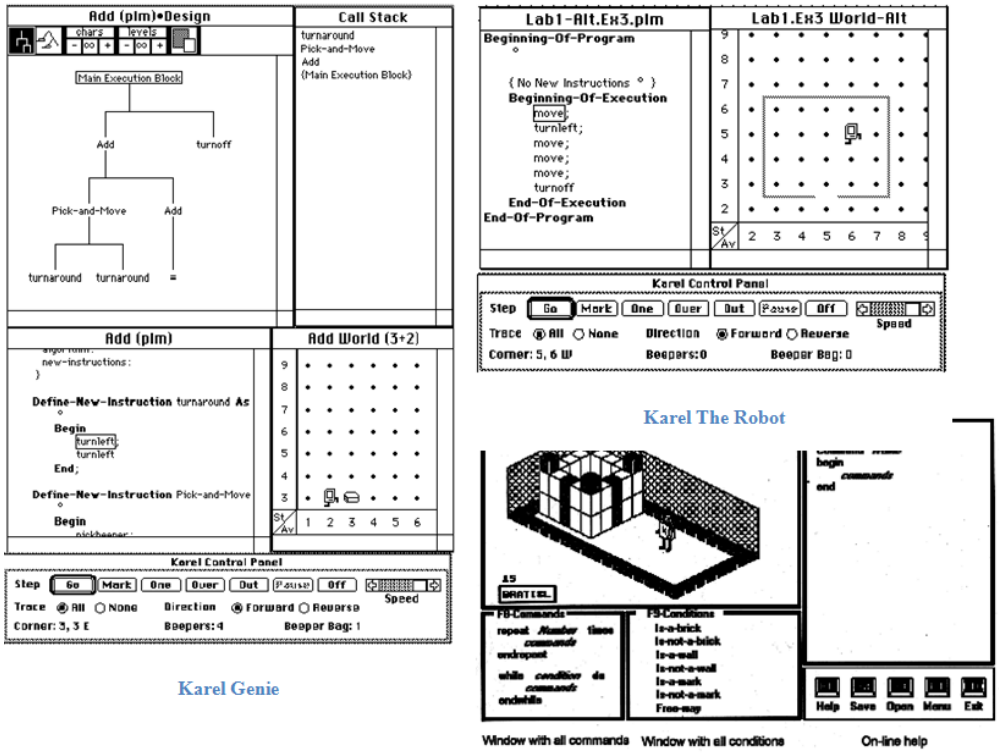

Figure 1: Variants of Karel

Some work has already been done in this regard, e.g. (Linda Grandell, 2006) and (Kevin R. Parker, 2006) discuss the process of choosing the first programming language to be taught to the students. Similarly, the customizations of the contents of the language for the first programming language are also discussed. In (Milbrandt, 1993) the authors emphasize on teaching with the help of problem solving skills. Some (Shaffer, 1986) (Milbrandt, 1993) have highlighted the importance of using visually interactive programming languages for the novice programmers.

3) PROPOSED QUALITATIVE FRAMEWORK

1) Simple syntax.
2) No pre-requisites.
3) Easy to implement.
4) Scripting programming language.
5) Easy transition to higher level languages.
6) Closely aligned to the curriculum.

3.1) Simple Syntax

One of the hard challenges for a new programmer is to become familiar with the syntax of the language in which she is programming. A simpler and clear syntax will surely help a novice programmer to grasp the language quickly. Especially, in our case, where we are concerned with very young students, simplicity and clarity is an inherent requirement so that the language should be easily understandable. We define complexity of the syntax in terms of inclusion of the header files, declaration of variable with data types, punctuation marks needed to write the code, and total number of lines of code.

Most of the programming languages taught at the high school level as a first programming language pose the problem of learning difficult syntax. Some need to include important file and lots of parenthesis and other punctuation marks which are hard to learn in the beginning, especially for younger students. The most common examples of the syntactic mistakes committed by the novice programmers are missing parenthesis, missing a semicolon and similar syntax errors.

The complexity in the syntax may frustrate the students and a novice programmer may lose her interest in programming. Similarly, a simpler syntax will surely help developing the interest in computer programming. Therefore, as a prime quality metric for our proposed framework we suggest that the language should have simpler and cleaner syntax.

Below, we present a comparison of our proposed simple language with Microsoft Visual Basic (Table 2).

Table 2 (Code Comparison)

Proposed language	**Visual BASIC**
1) a =10 2) b= 15 3) c = a+b 4) print c Run	1) Private sub button_click() 2) Dim a as integer 3) Dim b as integer 4) Dim c as integer 5) a = 10 6) b = 15 7) c = a+b 8) msgbox (c) 9) End sub

In Table 2, we have presented the code for a program that involves simple addition of two integer variables and displays the result. In the upper section we have presented the code for the same program in Microsoft Visual Basic language, which is claimed to be the best language for the beginner programmers (Kevin R. Parker, 2006). We can see that using Visual Basic this simple program is written on 9 lines; on the other hand, we have presented a sample code within four lines for the same purpose in our proposed language. This takes fewer lines and has much simpler syntax. Therefore, the example in Table 2 shows that by simplifying the syntax of the language we can save novice programmers from a lot of common mistakes.

Here, we can see that our proposed language has much simpler syntax which can be easily understood by a young student. So, in the beginning the students will write simple and easy codes. Whereas, as they progress, more programming constructs will be taught to them and they will be able to use those constructs to write more sophisticated code. We shall cover this point with more details in the coming sections. We claim that by bringing this simplicity in terms of writing the code we shall be able to increase the interest of the young students towards computer programming.

This simplicity in the syntax of the language requires a serious amount of work done. It demands significant changes in the compiler for its parser, linker and code generator. This is evident from Table 3, where we have presented the few constructs that become optional. Therefore, these changes require features like implicit assumptions about the data types,

acceptance of implicit declaration of a variable, and some other simplifications in the coding standards.

Table 3 (Comparison Result)

Constructs	Action
Semicolon (;)	Optional
Dim	Optional
As	Optional
Data type(integer)	Optional
Private sub	Optional
End sub	Optional
Msgbox	Replace with "print" statement.

3.2) No Pre-requisites

A serious consideration while designing a programming language for high school students is that these students do not have any prior knowledge of computer programming. Therefore, a very important quality of our proposed framework is the assumption that the language should not have any pre-requisites. This will help in designing a language which will be easier to understand, especially for new students.

However, a few simple requirements to learn programming in this new language will be the understanding of basic arithmetic and some understanding of English language. Learning mathematics and English language are already a part of the curriculum, so we can say that learning to program in a language that conforms to our proposed framework does not require any explicit pre-requisites.

3.3) Easy to Implement

One of the major concerns in terms of introducing a programming language at the high school level is difficulty in learning and implementing a programming language. This has already been a concern at university level as well (Mitchel Resnick, 2009). Therefore, naturally, the concern is if the high school students will have the capabilities to learn such languages. A few main problems faced by novice programmers in terms of implementation are use of wrong language constructs, passing the parameters etc. (Linda Grandell, 2006).

Hence, we propose that the language should be simple and many constructs should be made optional to simplify the implementation by a novice young programmer.

In order to make a language simpler, we suggest the following amendments:

- The constructs like semicolon, parentheses block indentation of code and header file etc. can be made optional.
- More than one statement in proposed programming language will be considered as a block. In order to start a new block we can use "tab" as does the Python programming language. In the same "tab" all statements are considered in same block. So the programmer only has to use "tab" for selected line of statement in order to make it indented. This will also make the code more readable. This formatting of the code is one of the highly desired features in the software industry as well as it makes the maintenance of the code easier and also helps in extending the code more easily.
- Hide all the constructs that make a hurdle in terms of understanding due to lack of relevant knowledge. This involves alignment of the programming language course with the syllabus of other relevant subjects.

This requires the design of a sophisticated Integrated Development Environment (IDE) tool. Such tools are widely used at professional level as well, and help the programmers implement the code easily. Similarly, at such novice level such a tool will ease job of a programmer. We strongly believe that an IDE tool with rich features list is one of the major requirements to make the implementation easier.

3.4) Scripting programming language

We argue that the proposed language should be a scripting language and it should not be an imperative programming language. The reason is that scripting programming languages are not ideal for system programming, and we cannot use them as operating system programming language. Also the scripting languages are especially designed for a quicker feedback to the user, e.g. they are widely used in Web based applications and many desktop applications. More importantly, it has been established

in (Mendelson, 1990) that the novice programmers ask for a quick and fast feedback.

Scripting programming languages like python, php, and VB are different than other programming language. These kind of programming assumed that building block of development already have been developed and in scripting language just need to use no need to develop it from scratch. Scripting language normally used for enhancement of features of complex algorithm and data structures. Such features like usually provided by component. Scripting language some time called system integration language.

Secondly, it is important as a first programming language for novice programmers because in order to work in scripting language normally demanding less effort to write less code and get more output. Mostly plugins are available and can be customized according to need of the programmer. There are several valuable benefits of choosing a scripting language for the novice programmers. Firstly, they are easier to learn and write code. Secondly, the existing off the shelf components are available in the form of plug-ins and they can be seamlessly integrated in such languages. "As far as I can tell, the way they taught me to program in college was all wrong. You should figure out programs as you're writing them, just as writers and painters and architects do" (O'Reilly, 2003). This again highlights the need of an environment which makes computer programming easier that can be achieved by a simple and easier IDE.

3.5) Easy Transition to Widely used Higher Level Languages

The most common agreement in all research centers related to computer programming is that the first year programming language for a novice programmer must be a high level language. The idea is that the first ideal programming language for high school should be a subset of a high level programming language. This can be any high level programming language which is widely used in educational institute for the initial programming courses. It can be **C/C++** as **MINI C** or **JAVA** as **MINI JAVA** or any language that is meets above criteria**.** Another advantage of choosing such a language which is widely used in academia is that there is a lot of supportive material such as tutorials, resources, books and trainers of that language.

Besides the contents and resources of programming language, the high level programming languages are close to human understanding as compared to a low level language. Especially, for the beginners it is necessary that they should learn high level programming language as their first year programming course (SeungWook Yoo, 2006).

Moreover, the main purpose of introducing programming language at high school is to learn the language constructs and programming logic along with the other relevant courses. This will help in improving the understanding of the programming language, and will also improve the thinking level of the students. Writing a piece of code demands a clear understanding of the problem, therefore, writing the program for a certain problem will inevitably improve the understanding of the problem and its domain. It will also help the students their logic building and reasoning skills used behind any action of computer during their interaction with it. So, it is important to highlight a point for the student that learning the programming skills will be a very useful tool in their personal skills from which they can benefit at any stage of their lives. This will make the whole exercise of learning computer programming very easier and effective as it involves a natural phenomenon of introducing a new thing to the human which draws their attention in many ways.

3.6) Closely Aligned to the Curriculum

A very important consideration while designing a programming language for high schools students is to introduce the concepts in a smooth and gradual manner. This involves a serious task of examining the other relevant courses and identify the contents of those courses which are required to learn how to program. The most important course in this regard is that of Mathematics. Therefore, the course of programming language should be synchronized with the course of mathematics, and no constructs from the programming language should be introduced to the students prior to the corresponding concept in mathematics. This will also help the students in solving the problems/ exercises of other course work using computer programming. The student will benefit from this activity and will be able to understand the use of software and computer programming in various different fields. This will not only increase the student's interest in computer programming, but will also be beneficial in better understanding of the concepts of other subjects.

As an example, let us consider a simple mathematical problem which computes the average of two given numbers. Table 4 shows the solution in mathematics and its corresponding implementation in our proposed simplified high school programming language. We can see that our proposed programming language is very close the mathematical expressions, thus, it involves lesser effort in terms of learning the syntax of the code. It is important to note that this simplified syntax is different from the syntax of existing languages, as shown in Tables 2 and 4. However, this needs to make several changes in the design and implementation of this programming language. Intuitively, it is clear that the compiler of an existing high level language needs to be modified to incorporate the simplified syntax, as discussed in this paper.

Table 4: (Code Comparison)

Solution in Mathematics	**Solution in Computer Program**
Number1=10 Number2= 20 Average = (Number1+ Number2)/2 Answer = Average, which is 15	Number1=10 Number2= 20 Average = (Number1+ Number2)/2 Print average.

In our proposed language a high school student can develop a sense of writing algorithms, create logic, do some reasoning for the problem at hand, which will be beneficial for her later in her life in many fields in several ways. Whereas, strict adherence and synchronization to the syllabus of other relevant courses helps in terms of teaching and introducing the programming constructs over a timeline.

Another important factor in designing the curriculum is to provide useful text books for every school grade. Each books should contain description of the topics, examples related to that topic, problem exercises for these topics. Therefore, designing thorough practical and effective text books is also holds an important position in this project.

4) DISCUSSION

4.1) Technical Requirements

The idea presented in this paper is simple yet important; however, it requires a big amount of work to be done in terms of tools and technologies. Technically, our proposed project affects the syntax and semantics of the existing programming language. Our requirement is that the compiler should understand both existing programming language, as well as, our proposed simpler programming language syntax. This needs major changes in terms of theoretical design of the language, and all parts of the compiler, namely, the lexical analyzer, syntax analyzer, and the code generator, which is very time consuming process. Alternatively we can design a preprocessor while will implement lexical, syntax and semantic analyzer and produce intermediate code in some high level programming language which should be compatible with existing compilers.

The other important aspect is the development of a supporting Integrated Development Environment (IDE) for this purpose. This IDE should be equipped with strong features to help the young learner in coding and debugging. For instance, the IDE should have support to include the required library files, based on the code. Another attractive feature, especially, for the young beginner students, can be the writing of code with the help of some UI features, e.g. buttons. The student can write code for declaring variable, display message box, and many other most frequently used operations. The IDE should generate the code based on user's commands. This way, in the beginning, the young programmers can write the code with the help of the User Interface (UI). However, later on, as the student progresses, the IDE should raise the level of difficulty, and the programmer should write the code herself. This way the IDE can be used for examination purpose as well.

Apart from the tools and technologies, writing and preparing appropriate syllabus books for each distinct grade for high school also demands serious deliberation. Proper text books with all necessary ingredients including theory, examples, and exercises are core requirements for the success of this project. A proper road map is required for this purpose, which defines all the topics to be covered in each year. This in turn,

requires another level of synchronization of the teaching material with other relevant courses, mainly, mathematics for concepts, and in general with other related courses for example scenarios and exercises. Finally, thorough reviews for the curriculum are required by the educationists to approve the quality of the designed curriculum and books.

4.2) Expected Outcome of the Project

In general, such programming language courses are taught with the following objectives:

1) The student should develop critical and logical thinking.
2) The students should possess problem solving skills and should be able to understand the problem, and they should be able to design, implement and debug it.
3) The reluctance of the students towards writing code should be eliminated. After taking such courses the students should be able to write the code without any hesitation. This is affected by not only the language and available tools, but is also affected by the teaching methodology. This will make the student comfortable with computer programming in future (Linda Grandell, 2006).

Therefore, a major contribution of our research is the intention to produce students who possess the above mentioned objectives. Secondly, we want to develop supportive tools and technologies including the preprocessor; a rich IDE, mainly focusing at the ease in coding; and appropriate text books to help the instructors and students make this a success.

5) CONCLUSION AND FUTURE DIRECTIONS

In this paper we have proposed a qualitative framework for designing a programming language course for the high school students. The proposed framework is a comprehensive effort which encompasses all major aspects of the problem, in terms of tools and technologies; and in terms of educational requirements. Above all, this work lays a solid foundation for the development of high school level programming course.

This work opens several future directions based on a number of major tasks identified during this phase; namely, design of the language and implementation of its compiler. Another important activity is the design

and development of curriculum. Lastly, design and implementation of a rich Integrated Development Environment is necessary for the accomplishment of the whole project.

REFERENCES

Calabrese, E. (1989). Marta - the "Intelligent Turtle". *Proceedings of Second European Logo Conference, EUROLOGO'89* , (pp. 111-127).

Chen, X. N. (2004). First Programming Languages Revisited. *College Teaching and Learning Conference*, (pp. 27-32). Orlando, Florida.

Clancy, M. (2004). Misconceptions and attitudes that interfere with learning to program. *Computer Science Education Research.* (pp. 85-100). London, UK: Taylor and Francis Group.

David Smith, P. L. (2010). Using educational programming language to enhance teaching in computer science. *Edge Conference.*

Elkner, J. (2001). Using Python in a high school computer science program. *Ninth International Python Conference.*

Hvorecky, J. (1992). Karel the Robot for PC. *Proceedings of East-West Conference on Emerging Computer Technologies in Education* , (pp. 157-160). Moscow.

Kay, J.A. (1993). A microworld for developing learning design strategies. *Computer Science Education 3 (1)* , 111-122.

Kevin R. Parker, J.T. (2006). A Formal Language Selection Process for Introductory Programming Courses. *Journal of Information Technology Education*.

Li-Chun Wang, M.-P. C. (2010). Learning Programming Concepts through Game Design: A PCT Perspective. *DIGITEL - Digital Game and Intelligent Toy Enhanced Learning.*

Linda Grandell, M. P.-J. (2006). Why complicate things? Introducing programming language in high school using Python. *Australasian Computing Education Conference - ACE .*

Mendelson, P. G. (1990). Programming languages in education: the search for an easy start. *Psychology of programming*, 175-200.

Mendelson, P.G. (1990). *Programming languages in education: the search for an easy start. In Psychology of programming.* London: Academic Press.

Milbrandt, G. (1993). Using problem solving to teach a programming language in computer studies. *Journal of Computer Science Education 8(2)*, 14-19.

Mitchel Resnick, J.M. (2009). Scratch: Programming for Everyone. *ACM.*

Olimpo, G. (1988). The Robot Brothers: An environment for learning parallel programming oriented to computer education. *Computers and Education*, 113-118.

O'Reilly, T. (2003, May 14). *Why Scripting Languages Matter*. Retrieved January 12, 2012, from oreilly.com:http://oreilly.com/pub/wlg/3190

Papert, S. (1980). *Mindstorms, children, computers and powerful ideas.* New York: Basic Books.

Pattis, R. E. (1981). *Karel - the robot, a gentle introduction to the art of programming.* London: Wiley.

Peter Brusilovsky, E. C. (1998). Mini-languages: a way to learn programming principles. *Education and Information Technologies 2, 1*, 65-83.

Programming at high school. (2010, 10 12). Retrieved January 2012, from www.techrepublic.com:http://www.techrepublic.com/blog/programming-and-development/how-to-introduce-high-school-students-to-programming/1511

Schollmeyer, M. (1996). Computer programming in high school vs. college. *SIGCSE '96: Proceedings of the 27th SIGCSE technical symposium on CS education*, (pp. 378-382).

SeungWook Yoo, K.-A. K. (2006). Empirical Study of Educational Programming Language for K12. *IJCSNS International Journal of Computer Science and Network Security* , Vol. 6 No. 6.

Shaffer, D. (1986). 'The use of logo in an introductory computer science course. *SIGCSE Bull. 18(4)*, 28-31.

Xiaoxia Wang, Z. Z. (2011). The research of situational teaching mode of programming in high school with Scratch . *ITAIC - IEEE Joint International Information Technology and Artificial Intelligence Conference.*