Socratic Programming: An Innovative Programming Learning Method

Imad H. El-Zakhem

***Abstract*—In second language acquisition (SLA) environment, learner is a community member and communicates with others, whereas, in a foreign language learning, the practicing dimension is weaker. When it comes to programming, there is no community using a programming language, thus a SLA environment cannot exist. For this reason the most adopted way is teaching programming like a foreign language with all its drawbacks. In this paper we compare between programming language and natural language, knowing that the programming introductory courses have basic concepts without complex algorithms, in other terms more expressive than computational. We propose a new method of teaching programming based upon dialogues between a facilitator and students. The facilitator will be part of a communication between each student and the computer to make an environment suitable for a SLA. A Socratic way of learning is achieved with a teacher having few students and can participate with his students in their dialogue and helps through their guidance to express their ideas. We list 4 common programming problems and we showed that using dialogue and raising questions, students were able to resolve these problems.**

***Index Terms*—Memory concept, programming language, programming learning, Socratic dialogue.**

I. Introduction

Programming has been always seen as a difficult discipline to learn and teach.Reference [1] reported that the rate of failure or drop in undergraduate programming courses varies from 25 to 80 %. The importance of the computer as an analysis or simulation tool in the experimental sciences makes the problem of introductory programming course a weighted issue. A synthesis of various studies of psychology and educational programming shows the different difficulties in learning programming. According to [2], students are not well oriented and don’t know what programming is about also the computer is seen as a notional machine or as a black box [3]. The anthropomorphical errors are typical errors and are committed when the student assumes that the computer has a hidden intelligence and it is expected to understand what the student has in mind [4]. These difficulties are faced when using a classical approach in teaching and using a general purpose language and a professional environment according to [5] and [6]. According to ACM, the introductory course should include only basic concepts of programming without tackling complex algorithms [7]. At the level of resolving primitive problems, the use of a programming language is similar to the use of a natural language to a far extent. In this paper we will compare natural language to programming language in Section I, highlight the fact of learning a new language in Section II and describe the appealed difficulties in Section IV.The results of a study conducted are shown in Section V and a new approach in learning programming is emphasized. Section VI concludes this paper.

Manuscript received March 25, 2014; revised October 27, 2014.

Imad H. El-Zakhem is with the Department of Computer Science, University of Balamand, Lebanon (e-mail: izakhem@balamand.edu.lb).

II. Natural Language and Programming Language

A natural language is a language used by humans to communicate. A programming language is a language used by humans and computers to communicate. This is the reason it is being called a programming *language*.

There are many natural languages and every country has its native language. Similarly there are many programming languages used by different scientific communities.

Natural languages have properties and concepts which are common between all languages.The nouns and verbs and the different grammar properties are the same in all languages. We have always the past tense, the plural, the feminine and masculine, etc. The Universal Grammar proposed by Chomksy describes a common formal structure among all natural languages [8].

While natural languages are characterized by their vocabulary and grammar, programming languages have their syntax and keywords. In many cases, a simple program is just a translation from the natural to the programming language.

While natural languages are much more complicated and fuzzier, programming languages are more formal and structured. Fuzzy theory proposed by Zadeh is a tool to bridge between the two types of languages [9]. Using fuzzy set theory the representation of many concepts in natural languages is feasible by using computational terms through a programming language [10].

III. New Language Learning: Foreign Language Learning vs. Second Language Learning

Foreign language learning terminology is used to describe the fact of learning a language which is different than the society language, for example, learning English in Russia. Second language learning describes the fact of learning a language in a foreign country. For example a Russian who is learning French in France. In the case of second language learning, the term acquisition is more used since the learner is acquiring the language because he is living in a foreign environment and is learning through this environment. The term *second language acquisition* has replaced the term *second language learning*.

When learning programming by instructions and lectures we are dealing with the programming language like a foreign language. The difficulties raised during this process are described in [11]. In the other hand, the second language acquisition is smoother, faster and more fruitful [12].

IV. Methods and Difficulties in Learning a New Language

Learning is a lasting change in the cognitive structures of the individual. Information (verbal, visual, auditory, motor) is stored in memory in the form of nonlinear networks. The network developed about a concept is unique to the person; it depends on his own history. Learning is the result of the transformations occurred upon the individual activity like seeking information, analogies, discrimination, hypothesizing [13]. Learning is also described as an individual active construction that requires an emotional investment. The sustainability of change achieved may depend on repetitions, on the variety of channels used for transmitting information and on the depth of processing performed by the learner [14]. The constructivist approach tends to embed learning in realistic and relevant contexts, encourage ownership and voice in the learning process; students identify their goals and objectives, embed learning in collaborative social experience [15].

The teacher should promote the retention of learning by creating mental images, facilitating an initial understanding of the concepts in greater proximity to the semantic network of the learner and helping with the information processing.

Two major difficulties in language learning are the anxiety and the attitude of the learners.

- Anxiety is shown to be a major obstacle of learning a language. Learners have fear of being misunderstood and feel threatened by using an unknown communication tool [16].
- The attitude and the motivation are psychological factors that have also great influence on foreign language learning. There are two types of motivation, integrative motivation which describe the willingness of a learner to be a member in the foreign community and instrumental motivation which describes the desire of a learner to seek recognition from the whole learning process. [17], [18]. High rates of accomplishment are witnessed with motivational rather than instrumental motivation [19].

V. Socratic Methodology for Programming

In a SLA scenario, the learner should live in the foreign country, take some classes and most important of all communicate with the society. The lack of communication will get him back to the scenario of foreign language learning.

While in programming language, an excessive communication with the machine using the programming language is equivalent to SLA. Furthermore, the instructor will play the role of a facilitator who will assure the viability of communication. Since the learners couldn't communicate with each other with the programming language but only with the computer, the instructor should facilitate the dialogue between each student and the computer, thus participating in many conversations simultaneously. Each dialogue has three parties, the facilitator, the learner and the computer. The Socratic methodology consists of raising questions and dialogue between participants. Socratic dialogue, as we are told by Plato, aims to bring the disciples to question their opinions and beliefs. Socrates requests justifications in the form of rational arguments from his partners and aim to bring them gradually to discover what they already know and to become sure about the truth and virtue.

Below are four examples of successful dialogues which led students to write consistent and correct programs. 90 sophomore students are asked to solve the following problems in their first introductory programming course.

A. The First Example Shows Students the Correlation between Human Memory and Thinking on One Side and Computer Memory and Processing on the Other Side

- Problem 1: Askthe user to enter two numbers, obtain the numbers and display the sum.

According to human reasoning:

Ask the user ⇔ Display a message.

Obtain 2 numbers ⇔ Store the numbers in memory. For this reason, memory should be ready to receive the values and that what the declaration of variables stands for.

Display the sum ⇔ Access the values already stored in memory and make the addition. If the numbers are forgotten the addition turned to be impossible. Thus storing the values of variables in memory is crucial both in real life and in programming in order to achieve the task. The verb *remember* is equivalent to *store value in a variable already created in memory*. Memory in Programming is Similarto how human hsestheir memory's neurons and synapses, and in simple cases, writing a program is achieved by expressing what is happening in the human memory.

A pseudocode of the program is as follows:

```
Declare var1, var2
Ask user to input
Input values into var1 and var2
Display var1 + var2
```

The memory concept is primitive for programming and it is not tackled in mathematics or other sciences; furthermore the notions from mathematics is hardening and confusing learners: For example, the notion of variables in programming differs from that in mathematics; variable in programming is a memory location which holds any value and can change its content, while a variable in math can hold any value but cannot change its value once set. In programming, a value of a variable is an existing value for an existing object in memory and can be accessed and modified while the variable in mathematics is totally abstract.

B. A Second Common Exercise Using Variables in Programming is the Swap

- Problem 2: Having two variables x and y with values 3 and 5 respectively, swap the values of the 2 variables.

Only 7% of the students achieved their goal. The statements $x = y$ and $y = x$ was the response of 76% of the students. When the students were asked how to swap the containing of two cups, one with water and one with oil, all students agreed that a third cup is needed to hold temporally the water and achieve the swap.

A pseudocode of the program is as follows:

```
Declare x = 3, y = 5
Declare Temp

Temp = x
x = y
y = temp
```

Students were not able to resolve the problem by using abstract mathematical variables, but when they think that a variable is a physical container or a memory location, the solution becomes obvious.

C. Maximum of Numbers

- Problem 3: find the maximum of 4 positive numbers.

78% of the students tried to write *nested if-else* statements and 10% wrote *if* statements with logical operators.In order to approach a rightful method to resolve the problem we changed the setting and the instructor played a game with one student and asked the other students to describe how their peer is going to find the right solution. The instructor gave a sequence of numbers to their peer and at the end the student was able to give the right result by memorizing the maximum number and comparing each new number to the memorized maximum. After the game, 85% of the students were able to write the following pseudocode:

```
Declare m, no1, no2, no3, no4

Get no1
M= no1
Get no2

If (no2 > m) m = no2
Get no3

If (no3 > m) m = no3
Get no2

If (no4 > m) m = no4
Display m
```

With a more in depth look to the above code, 22% of the students can optimize the code and use only one variable to get the values and one variable to store the maximum and to repeat the steps.

```
Declare m, no = 0

Repeat 4 times
    Get no
    If (no > m) m = no

Display m
```

The above optimized code is achieved when the students are asked if they remember any of the values except of the maximum.

D. Adding up 5 Numbers

- Problem 4: Ask the user to get 5 numbers, add up the numbers than display the result.

95% of the students declared 5 different variables and a variable *total* to hold the total. They wrote a program to input the values into the variables and add up these values using the variable *total*. When asked to do the same problem but with 100 variables, most of the students were able to understand that a loop is needed but they faced difficulties in how to declare variables to store the 100 values. 21% of the students wrote a code similar to the code below:

```
Declare no, total
Repeat from 1 to 100

Input a value into the variable no
total = total + no
```

The rest of the students just wrote the loop header but struggled to achieve the input and the sum. A discussion occurred about how the problem can be resolved mentally. The students can describe the solution in English as follows:

```
Repeat the below steps100 times:
    Get a number
    Add it to the total
    Remember the total for the next iteration
```

At the end of the problem the given values are overwritten and only the total is kept in memory. A comparison between the description written by the students and the correct pseudocode tells us that with little help those students will be able to resolve the problem.

In the four examples above, the description of the solution is translated from English to programming. Students were able to develop the right solution when first they assimilate the memory concept and second they express their ideas.

VI. Conclusion

Programming languages seems to be difficult at the beginning but the learner becomes more fluent after practice, like in all natural languages.

The questions raised and the dialogue between facilitator and students ensure that the students will find the solutions and they will be independent from classic tutoring. They become more motivated and they develop a mental model about programming concepts.

Using a kind of conversation between few participants is feasible because there are no complex algorithms or problems to be resolved in the introductory programming courses. The formal lectures shall be shorter in time and learning time is adjusted according to the level of fluency, the number of participants and the difficulty of the subject in question.

References

[1] J. Kaasboll, *Learning Programming*, University of Oslo, 2002.
[2] B. D. Boulay, "Some difficulties of learning to program," *Studying the Novice Programmer*, Lawrence Erlbaum Asssocites, pp. 283-299, 1989.
[3] M. Ben-Ari, "Constructivism in computer science education," in *Proc. 29th ACM SIGCSE Technical Symposium on Computer Science Education*, Atlanta, Georgia: ACM press, 1998.
[4] J. Spohrer, "Analysing the high frequency bugs in novice programs," presented at First Workshop on Empirical Studies of Programmers, 1986.

[5] P. Brusilovsky, E. Calabrese, J. Hvorecky, A. Kouchnirenko, and P. Miller, "Mini-languages: A way to learn programming principles," *Education and Information Technologies*, vol. 2, pp. 65-83, 1997.

[6] S. Xinogalos, "Educational technology: A didactic micro-world for an introduction to object-oriented programming," Ph.D. Thesis, Dept. of Applied Informatics, University of Macedonia, 2002.

[7] The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society. (2013). Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. [Online]. Available: http://www.acm.org/education/CS2013-final-report.pdf

[8] N. Chomsky, *Aspects of the Theory of Syntax*, MIT Press, 1965.

[9] L. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning," *I-III, Information Sciences*, vol. 9, pp. 43–80, 1976.

[10] L. Zadeh, *Computing With Words. Principal Concepts and Ideas*, Berlin: Springer, 2012.

[11] W. Lambert, "Psychological approaches to the study of language," *Modern Language Journal*, vol. 47, pp. 51-62, 1963.

[12] J. H. Hulstijn, "Fundamental issues in the study of second language acquisition," *EUROSLA Yearbook 7*, pp. 1569-9749, John Benjamins Publishing Company, 2007.

[13] J. Elman, E. Bates, M. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett, *Rethinking Innateness: A Connectionist Perspective on Development*, Cambridge, MA: MIT Press, 1996.

[14] M. Daly, *Developing the Whole Child: The Importance of the Emotional, Social, Moral and Spiritual in Early Years Education and Care*, Wales: Edwin Mellen Press, 2004.

[15] D. Cunningham, T. M. Duffy, and R. Knuth, "Textbook of the future," in *Hypertext: A Psychological Perspective*, C. McKnight, Ed., London: Ellis Horwood Publishing, 1993.

[16] S. Kirova, B. Petkovska, and D. Koceva, "Investigation of motivation and anxiety in Macedonia while learning English as a second / foreign language," *Procedia - Social and Behavioral Sciences*, vol. 46, pp. 3477-3481, 2012.

[17] R. C. Gardner and P. C. Smythe, "On the development of the attitude/motivation test battery," *Canadian Modern Language Review*, vol. 37, pp. 510-525, 1981.

[18] A. Masgoret and R. Gardner, "Attitudes, motivation and second language learning: a Meta-analysis of studies conducted by Gardner and Associates," *Language Learning*, vol. 53, no. 1, pp. 123-164, 2003.

[19] W. Lambert, "Persistent issues in bilingualism," in *The Development of Second Language Proficiency*, B. Harley, P. Allen, J. Cummins, and M. Swain, Eds., Cambridge University Press, 1990, pp. 201-218.

Imad H. El-Zakhem holds a Ph.D degree in computer science from the University of Reims, France, and a master degree in electrical and computer engineering from the National Technical University of Athens, Greece. The major fields of study are the perception modeling and programming learning methodologies and difficulties in programming. He is currently an assistant professor in the University of Balamand and served as an IT manager and a project developer in private companies. He has many publications in color perception and image classification using fuzzy set theory and in methodologies of learning programming, learning programming in schools and difficulties of learning programming.