Understanding the programming variable concept with animated interactive analogies

DOUKAKIS D., GRIGORIADOU M., TSAGANOU G.

Abstract---- **One of the difficulties that novices face in introductory programming courses in high school and universities has to do with understanding the concept of programming variable. Student's prior knowledge is related with the concept of the variable as it is defined in mathematics. This knowledge comes in conflict with the new knowledge for the programming variable.**

The first part of this paper examines whether the misconceptions that students have on the programming variable can be explained by their prior knowledge about mathematical variable, and whether understanding of the programming variable concept can be considered a conceptual change case where a radical reorganization of their existing knowledge structures is needed.

In the second part an instructional intervention with a learning object that included an animated interactive analogy is compared with an instructional intervention using a learning object without the analogy.

***Index Terms*-- Analogies, Interactive animation, conceptual change, introductory programming**

I. INTRODUCTION

It is a common place for introductory programming teachers that most of the students face serious difficulties in understanding the important concept of the programming variable as defined in procedural strongly typed languages like Pascal, and QBasic. Various studies [12], [3], [11], [4], [14], [15], [6] support teacher's anecdotal observations with empirical data and give detailed reports on the misconceptions students have concerning the programming variable.

Some of these studies account for these difficulties and misconceptions instruction with inappropriate analogies that can be misapplied by students [3]. It is true that inappropriate analogies like the box or the plate analogy have been used by textbooks and teachers but problems seem to exist even in cases where no analogies are used.

Others account for these difficulties and misconceptions the transfer of problem domain knowledge into the programming domain [12]. For others [11], misconceptions appear to be the results of inappropriately bringing knowledge from algebra to programming.

A different way to look into this problem is also based on student's prior knowledge and follows the conceptual change approach [5]. The term conceptual change is used to characterize situations where learner's prior knowledge is incompatible with the new conceptualization and

Doukakis D. is PhD candidate, Interdisciplinary Program of Graduate Studies in Basic and Applied Cognitive Science University of Athens e-mail: d.doukakis@ sch.gr).
Grigoriadou M. is Associate Professor, Department of Informatics and Telecommunications, University of Athens (e-mail: gregor@di.uoa.gr).
Tsaganou G. is PhD, Research fellow, Department of Informatics and Telecommunications, University of Athens (e-mail: gram@di.uoa.gr).

where learners are disposed to make systematic errors or built misconceptions. Although there are many theories of conceptual change [16] there are some key points that are common to most theories of conceptual change. These key points are:

- In some learning situations the new knowledge cannot be added to the prior knowledge following an enrichment procedure.
- This is because the prior knowledge comes in conflict with the new knowledge.
- In these situations radical reorganization of prior knowledge is needed.
- In these cases students are more likely to face difficulties and have misconceptions.

II. Understanding the Programming Variable Concept as Conceptual Change Case

Students without prior programming knowledge, who come to the computer programming courses, know the concept of the variable as this is defined and used in high school mathematics. In mathematics a variable is a name (usually one alphabet letter) that can stand for numerical values or other algebraic objects.

The concept of variable in programming shares similarities with the mathematical variable but also has many important differences. The programming languages that are taken into account in this study are procedural strongly typed languages like QBasic and Pascal that are commonly used for introductory lessons. In these languages a variable is a symbolic reference to a computer memory location that can store values of a specific type. The variable values can be integer numbers, real numbers, logical values and alphanumeric values (characters and strings of characters). The values stored in a programming variable can change dynamically during the execution of a program. The variable name can contain both numeric and alphanumeric symbols but it has to comply with the naming conventions of each programming language. Usually the variable name denotes its use inside the program.

Although the same term “variable” is used in both contexts (mathematics and programming) the fact that we are referring to two different concepts is made explicit by most teachers and books.

Student’s misconceptions about the programming variable can be related with their prior knowledge about mathematical variable as shown in [5]. Some of these points are given in table 1.

Table 1 . Relation of students' prior knowledge from mathematics with their misconceptions in programming

Prior knowledge from mathematics	**Misconceptions in programming**
In mathematics there are no alphanumeric variables. All variables are of arithmetic type i.e. they take values from system of numbers like N, R etc. As a result students have no experience at all with the use of alphanumeric variables.	• Students think that they can use arithmetic operators (like +,-,/,*) on logical and alphanumeric variables. • They confuse the content of an alphanumeric variable with its name or the name of other variables used in the program.
In some exercises students are used in replacing an arithmetic formula with its equivalent variable.	They assign the symbols and not the computed value of the arithmetic formula when an arithmetic formula is assigned to variable (they think that the formula should be stored and not its value).
In mathematics, variables have only symbolic existence and not physical existence.	• Students think that programming variables are not stored inside the computer or they are unable to define their location. • Students are unable to define the initial value of a variable that has just been declared but no value is assigned to it yet.
In mathematics a variable can stand for an integer or real number with very big size or a real number with infinite numbers of decimal digits.	• Students think that there is no size limit to the values we can store in a programming variable. • Students think that there is no precision limit to the values we can store in a programming variable.
In equation solving the result can have the form of a double solution i.e. "X=1 or X=-3" where the equation is valid for more than one values of the unknown variable.	Students think that a programming variable can store many different values at the same time i.e. remember the history of assignments.
In mathematics problem solving students are used in using a variable first in calculations and define later the number system that it belongs to.	Students think that a programming variable of a specific type can also store values of different types

So in the case of the understanding of the programming variable concept by students, all the key points of conceptual change, described in introduction, are present:

- The difficulties students face in the understanding of the concept of programming variable are greater than the difficulties they face in the understanding of other programming concepts that doesn't come in conflict with their prior knowledge i.e. in the understanding of the operation of a read command [12].
- Student's prior knowledge about the mathematical variable is not sufficient for the understanding of the programming variable.
- The new knowledge cannot be added smoothly in the existing knowledge structures. This occurs because the prior knowledge about the mathematical variable comes in conflict with the new knowledge about the programming variable.
- For understanding the programming variable students need a radical reorganization of their knowledge structures. Therefore understanding the concept of programming variable can be considered a conceptual change case.

III. Instructional intervention using animated interactive analogies

Although in conceptual change cases students' prior knowledge causes problems in learning it can also proved to be very helpful. Students' prior knowledge can be used by instructors for selecting an appropriate analogy that will foster learning.

An analogy is a mapping between similar features of dissimilar concepts, principles or formulas [7]. When the analogy is used for instruction the known concept is called source (analogue) and the concept to be taught target. Ideally the source of an analogy used for instruction should be familiar to all students. A properly selected analogy should have a big degree of correspondence between the source and the target concept. Of course since the two concepts are not identical a perfect correspondence cannot be expected.

The analogies mentioned in the introduction have been used extensively to the instruction of programming variable. Usually these analogies are described orally by teachers or in written or image form in textbooks. A similar analogy for the programming variable is the mailbox analogy presented in students' textbook [1] with an image (figure 1). This analogy suffers the same misapplication problems with the box-variable analogy.

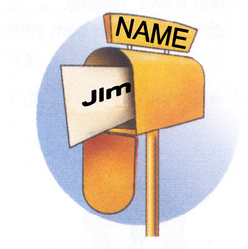


Fig. 1 : An analogy used for the programming variable

A different analogy for the programming variable was introduced in this study. It consists of a number of rotating cylinders with digits, characters or Boolean values depending on the type of the variable it represents (see figure 2). This analogy is animated and interactive to allow students' experimentation. It was expected to be familiar to the students from computer games (slot machine type) and television advertisements

In this study it is examined whether the provision and experimentation of students with the selected animated analogy can enhance understanding of the concept of programming variable. The analogy was embedded in a learning object i.e. an entity that can be used for learning. According to the definitions [17], [9] a learning object is small, independent, reusable in different contexts, and can be part of a bigger learning entity.

This learning object consists of six parts that students can visit in the order of their choice. The contents of each part are:

- Definition of the programming variable

- Programming variable types and examples that use the analogy
- Declaration of programming variables and example
- Syntax-function of read command and example that uses the analogy
- Syntax-function of value assignment command and example that uses the analogy
- Activity that uses the analogy. The activity follows the "Explorations" approach [10]. In this activity the students can enter an assignment command, enter values of their choice for the variables included in the assignment command, predict the outcome of the command and then "execute" the command to see the real outcome.

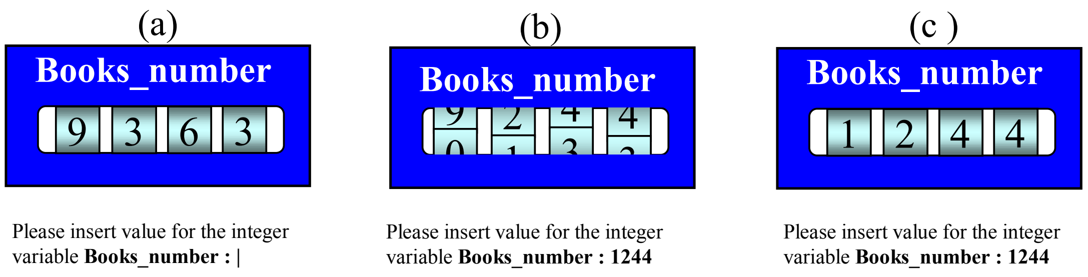
A pilot experiment was designed. The basic hypothesis of this experiment was:

- Will students using the analogy have an increased performance in questionnaire designed to measure their understanding and diagnose their misconceptions of the concept of the programming variable?

Secondary hypotheses tested in this experiment where:

- Will the total time students spend in learning object's parts that contained the analogy be increased?
- Will the total number of visits in the learning object's parts that contained the analogy be increased?
- Will the average level of interest for using the animated analogy be increased?

In this experiment GLOSSA [1] an educational Pascal-like programming language that follows the mini-languages approach [2] has been used. The value assignment operator used in this programming language is the symbol "🡐" and all reserved words are in Greek.



Please insert value for the integer variable **Books_number :** |

Please insert value for the integer variable **Books_number : 1244**

Please insert value for the integer variable **Books_number : 1244**

Fig. 2: Snapshots of animated analogy used for an integer variable (a) initial state (b) while cylinders are rotating after entering value and pressing enter (c) final state

IV. Method

A. Participants

Thirty-three introductory programming students from the last three grades of a high school (lyceum) in the Athens area participated in this study. Fifteen 1st graders aged 15-16 years and eighteen 2nd graders aged 16-17 years. Twenty-one were placed in the analogy group and twelve in the control group (the separation was done following the school schedule).

B. Materials

Two different versions of the learning object for the programming variable defined in the programming language GLOSSA [1], where used. The first version (analogy group version) employed the interactive animated analogy. The other version (control group version) used only a prompt to the student to give a value for a variable with a specific name and of a specific type ex." Please insert value for the integer variable **Books_number:**.". Then the value given by the student remains next to the prompt.

Navigation data logging capability was added to both versions of the learning object. The variables that where logged were the time spend in each part and the number of visits in each part of the learning object. A questionnaire (see appendix A) consisting of nine open type questions based on questions used in past studies [3], [14], [15], [6], was formed. The questions chosen were the more likely to reveal some of the misconceptions described in table 1. Two more questions where also added to indicate student's familiarity with the analogy (Q10), and interest level on a Likert scale (1-5) for using the analogue (Q11).

C. Procedure

Both groups studied the corresponding versions of the learning object without any time restriction. Then both groups answered questions Q1-Q9. Questions Q10 and Q11 where answered only by the analogy group. Questions Q1-Q9 were marked with 0 for wrong answers and 1 for right answers. Answers to questions that included explanations where considered correct only when both the answer and the explanation where correct.

V. Results

In order analogy to be useful for instruction it had to be familiar to students. Indeed a big percentage (83%) of analogy group students found the analogy familiar as indicated by the corresponding question (Q10). The differences in performance were in favor of the analogy group for all the 9 questions of the questionnaire (figure 3). Independents samples t-test revealed that the differences in performance were statistically significant for 5 of the questions (Q4, Q5, Q6, Q7, Q9) at a 0.05 level (table 2). These changes can be interpreted as indicating fewer misconceptions and better understanding of the programming variable concept especially in aspects where the control group was facing the greatest difficulties and misconceptions.

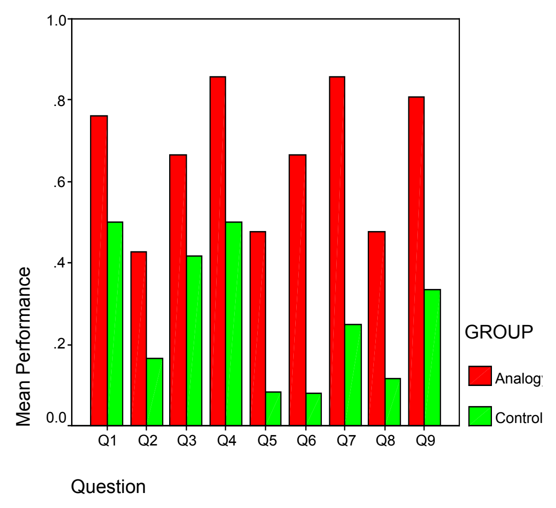


Fig. 3: Students'mean performance on questionnaire questions

Table 2 : Independent samples t-tests for analogy and control groups

Q1	T(31)=1,544, n.s.
Q2	T(31)=1,544, n.s.
Q3	T(31)=1,397, n.s.
Q4	T(31)=2,328, p=0,027
Q5	T(31)=2,436, p=0,021
Q6	T(31)=3,798, p=0,001
Q7	T(31)=3,521, p=0,001
Q8	T(31)=1,812, n.s.
Q9	T(31)=3,015, p=0,005

The analogy group students found the analogy "much interesting" (38%) and "very much interesting" (62%). There was also an increase in favor of analogy group in the total time spent on parts containing the animated analogy [*t(31)=2.329, p<.05*] and on number of students' visits on parts containing the animated analogy [*t(31)=3.826, p<.05*]. These changes indicate that students showed an interest on engaging in activities that employed the analogue.

VI. DISCUSSION

This pilot experiment cannot be used to draw any safe conclusions since the number of the participants was small and no qualitative analysis was done. Nevertheless it can provide some indications that the combined use of a properly selected analogy along with animation and interaction can help novice programmers deal with their misconceptions in the concept of programming variable.

Subsequent experiments with larger number of participants are planned to measure the individual effects of the factors that were used in combination in this experiment (analogy, animation and interaction). These experiment are planned to include qualitative analysis of the students' answers.

Analogies are well known for helping students understand complex science concepts but they are also well known for causing misconceptions in cases where students over generalize and map no corresponding features of concepts [13]. So an important question has to do with the problems that instruction with analogy is known to cause when carried to far [8]. In subsequent experiments questions will be added to detect novel misconceptions that can derive from misapplication of the specific analogy.

Another problem is that there is not always available an analogy familiar to all students that is also appropriate for the instruction of a specific concept. Therefore the above approach cannot be proposed as an instructional suggestion to all cases of conceptual change. However, is important to note that computer based simulations and animations give instructors the ability to bring in classroom analogies, that where difficult or impossible to use before.

REFERENCES

1] Bakali, A. Giannopoulos, I. Ioannides, C. Koilias, C. Malamas, K. Manolopoulos, I. Politis P. (1999), "Application Development in Programming Environment", Athens , Ministry Of Education and Religius Affairs – Pedagogic Institute.

[2] Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., and Miller, P. (1997) Mini-languages: A Way to Learn Programming Principles. *Education and Information Technologies* **2** (1), pp. 65-83.

[3] Du Boulay, B. (1989), Some difficulties of learning to program, In E. Soloway & J. C. Spohrer (Eds), *Studying the Novice Programmer*, Hillsdale, NJ, Lawrence Erlbaum Associates , 283-299.

[4] Ebrahimi, A. (1994), Novice programmer errors: language constructs and plan composition, Int. J. Human-Computer Studies, 41, 457-480.

[5] Grigoriadou, M. Kanidis, E. Doukakis, D. (in press) Students' beliefs on computer science concepts and the problem of conceptual change. *NOESIS (3)* Athens, Gutenberg: Typothito, (in Greek).

[6] Fesakis, G. Dimitrakopoulou, A. (2005) Cognitive Difficulties of Secondary Education Students relating to the Concept of Programming Variable, 3rd Panhellinic Conference "Didactics of Informatics", Korinthos, Greece.

[7] Glynn, S. M. Britton, B.K. Semrud-Clikeman, M. Muth, K.D. (1989) Analogical reasoning and problem solving in textbooks. In J.A. Glover, R.R. Ronning, & C.R. Reynolds (Eds.), "Handbook of creativity : Assessment, theory and research" pp. 383-398, New York:Plenum.

[8] Glynn, S. (1994) Teaching Science with Analogies: A strategy for Teachers and Textbook Authors, National Reading research center.

[9] IEEE Learning Technology Standards Committee (LTSC) (2001) *Draft Standard for Learning Object Metadata Version 6.1.* http://ltsc.ieee.org/doc/

[10] Lischner, R. (2001), Explorations : Structured Labs for First-Time Programmers, Proceedings of the ACM SIGCSE '01 Conference, Charlotte, USA, 154-158.

[11] Putnum, R. T. Sleeman, D. Baxter, J., Kupsa, L. (1989), A summary of the misconceptions of high school BASIC programmers, In E. Soloway & J. C. Spohrer (Eds), *Studying the Novice Programmer*, 301-314, Hillsdale, NJ, Lawrence Erlbaum Associates.

[12] Samurçay, R. (1989), The concept of variable in programming: Its meaning and use in problemsolving by novice programmers, In E. Soloway & J. C. Spohrer (Eds), *Studying the Novice Programmer, 161-178,* Hillsdale, NJ, Lawrence Erlbaum Associates.

[13] Thagard, P. (1992) Analogy, explanation and education. Journal of Research in Science Teaching, 29, 537-544.

[14]Tzimogiannis, A. Komis, B. (2000) The concept of variable in Programming: student's difficulties and misconceptions , in Komis, B, (eds) Procedings of the 2nd Panhellinic Conference "The Technologies of Information and Communication in Education", pp103-114, Patra, Greece.

[15] Tzimogiannis, A. Politis, P. Komis , B. (2005) Study of the Representations that Last Year High School Students have for the Concept of Variable, 3rd Panhellinic Conference "Didactics of Informatics", Korinthos, Greece.

[16] Vosniadou, S. (1999) Conceptual Change Research : State of the art and future directions. In Schnotz, W., Vosniadou, S. & Carretero, M. (Eds.) New Perspectives on Conceptual Change, (pp3-13). Elsevier Sciences Ltd.

[17] Wiley, David. A. (2002) "Connecting Learning Objects to Instructional Design Theory: A Definition, a Metaphor, and a Taxonomy." *The Instructional Use of Learning Objects* (Bloomington, IN: Agency for Instructional Technology)