Starting with Algorithmic Structures: A multi-representational, Real-life-activity learning environment for beginners

Maria Kordaki, Gregory Tsonis, John Palianopoulos and Aris Katis

Dept of Computer Engineering and Informatics University of Patras
e-mail: kordaki@cti.gr, tsonis@ceid.upatras.gr, palianop@ceid.upatras.gr, katis@ceid.upatras.gr

Abstract: This paper presents a multi-representational and real-life-activity based environment, the Starting with Algorithmic Structures (SAS) environment, designed for teaching the concept of algorithm and that of basic algorithmic structures to beginners. For the design of this environment, social and constructivist learning theories have been taken into account. By interacting within SAS, learners have opportunities to express their previous knowledge regarding the concept of algorithm and approach such concept through real-life activities. In addition, learners have the chance to express their algorithmic approaches to these activities using different Multiple Representation Systems (MRS) in a step-by-step learning experiment. In each step, constructive feedback is provided to the students to correct their actions. The MRS used are: natural language, flowchart and pseudocode. Interactive animations of the performance of these real life activities are also provided, inter-linked with the correspondent algorithm and its basic algorithmic structures in pseudo-code, for a better understanding of these structures by the learners. The features of SAS were pilotically evaluated with real students' whose positive effect on and their learning is also presented.

Keywords: Educational software, multiple representations, basic algorithmic structures, secondary education

Introduction

Understanding algorithmic logic is essential for K-12 students to grasp (ACM, 2003). This is based on the fact that appropriate algorithms play a crucial role in the design of programs aiming at the solution of real problems through the use of computers. Basic algorithmic structures such as; assignment statement, conditionals, iteration structures and recursion are of vital importance in the construction of these algorithms. Based on the above, paying attention to the understanding of these structures by the Secondary Education students is absolutely necessary. In terms of cognitive skills, the understanding of algorithmic logic demands the use of the learners' higher mental functions (Papert, 1980; Kurland, 1989). Understanding algorithmic logic presupposes students' ability to form a solution of the task at hand and to interpret this solution as a procedure in different representation systems such as: natural language, flowchart, pseudo-code and code in a specific programming language (Soloway & Spohrer, 1986; Jonassen, 1996). Students' difficulties are mainly attributed to the fact that they are not encouraged to master algorithmic logic in representation systems of various cognitive transparencies; starting from intuitive human oriented systems and moving gradually to more sophisticated machine-oriended systems (Christiaen, 1998). On the contrary, students are mainly introduced to the learning of algorithmic logic in the form of code using a specific programming language (Allwood, 1896). In addition, young students face difficulties in the understanding of algorithmic logic because this usually refers to problems presented in a mathematical-abstract form (Du Boulay, 1986; Soloway & Spohrer, 1989; Allwood, 1896). Such problems fail to serve any real purpose for the students. On the whole, students have difficulty understanding the concept of variables in programming as well as understanding all the basic algorithmic structures mentioned above (Samurcay, 1989).

A variety of educational computer environments are provided to help students learn about algorithmic logic and programming (Patis, Roberts & Stehlic, 1995; Freund & Roberts, 1996; Satrazemi, Hadjiathanasiou, Dagdilelis, 2000). However, there is a lack of such computer environments dedicated to teaching basic algorithmic structures to beginners providing: a) MRS supporting their step by step translation from their intuitive knowledge, expressed in human centered representation systems, to symbolic knowledge expressed in machine-oriented representation systems, b) step by step appropriate feedback and c) a motivating context consisting of every day life activities. Taking into account all of the above and in our attempt to support junior high school students to understand the

concept of algorithm and of basic algorithmic structures, we constructed a multi-representational, constructivist learning environment. This environment also emphasizes learning through experimenting with simulations of real life learning activities expressed in a diversity of representation systems. Such an environment has not yet been produced. In the next section of this paper, the design and the specific features of the proposed educational software are presented. Next, a pilot evaluation study of this software with real students is reported. Subsequently, the data from this pilot study are discussed and finally, conclusions and proposals for future research are drawn.

The design and the features of SAS

In our attempt to encourage young students to understand the concept of algorithmic and of basic algorithmic structures we constructed a computer learning environment namely: Starting about Algorithmic Structures (SAS). This environment was based on modern constructivist and social theories on learning emphasizing the active, constructive and subjective character of knowledge and the significant role of tools in knowledge construction (von Glasersfeld, 1987; Vygotsky, 1978). Constructivist computer learning environments are acknowledged as fields of primary reference to students' abstract knowledge (diSessa, 1987), as significant in the students' development of critical thinking (Papert, 1980; Jonassen, 1996; Noss & Hoyles, 1996) as well as intermediate worlds between the real world of concrete objects and the world of abstract concepts (Papert, 1980). In the context of SAS, the emphasis is put on the role of multiple representation systems (MRS). MRS provides students with possibilities for different starting points, so that they will be able to express their inter-individual learning differences (Dyfour-Janvier, Bednarz, & Belanger, 1987; Kordaki, Miatides and Kapsampelis, 2005). Multi-representational learning environments can also play a crucial role in the understanding of cognitively opaque concepts by the students. A plehora of such concepts are related to Computer Science and especially to computer algorithms (Kordaki, Miatides, Kapsampelis, 2005). In the context of SAS the emphasis is also put on students' active and practical involvement in their learning. To this end, simulations of every day life activities that are meaningful for the students are provided. These types of activities can encourage students to develop a strong motivation to be engaged in their learning (Nardi, 1996). Visual representations and immediate feedback on students' actions is also provided to help students reflect on them, correct their actions and form abstract concepts as well (von Glasersfeld, 1987). In fact, SAS was designed to provide students with opportunities to understand the concept of algorithm and of basic algorithmic structures through real life activities. In particular, students are provided with the opportunity to participate in a simulation of a simple every day life activity namely; the preparation of a cup of greek-coffee. In the context of this activity students have the chance to understand the description of this procedure as: a) a sequence of a definite number of concrete steps, b) a diagrammatic visual presentation, c) an appropriate natural language based recipe given to the computer to perform the real life activity at hand, d) as a recipe in pseudocode, which is significant as it implies the use of algorithmic logic.

Entering in SAS, each student is asked to give a login name so that all his/her actions with the software can be saved in appropriate logfiles. These files can be of use to teachers and researchers for further study and research. Next, some introductory remarks are provided to inform the learners about the objectives of this software and how to use its features. After this brief introduction, students have the chance to enter the main page of SAS. In this page, a colorful picture of a cup of greek-coffee is presented with a number of hyperlinks that lead the learner to the different parts of SAS. In each part a different representation of the procedure of the realization of the task at hand is performed. Moreover, in each part, students can express their previous knowledge about the specific sub-task at hand and also modify their knowledge by using specific help and immediate feedback. The general architecture of SAS is demonstrated in its main page that is consisted of the following parts-representations:

Part A: The concept of algorithm represented in natural language.
Part B: A real life procedure represented in algorithmic manner.
Part C: Commanding the computer to perform a real life procedure using natural language.
Part D: Real life procedures in the form of primitive programs in natural language.
Part E: Real life procedures in the form of primitive programs in Pseudocode.
Part F: Forming primitive programs in Pseudocode using basic algorithmic structures.

In the following section these parts are described in more detail.

Part A: The concept of algorithm represented in natural language

Step A.1: *Investigation of student pevious knowledge*. Here, students are asked to write in natural language what they believe an algorithm to be. The investigation of students' previous knowledge is important for teachers in

making appropriate decisions about teaching them about algorithmic logic and programming (Soloway & Spohrer, 1989). In the next step of the experiment, students can be involved in an assesment activity in order to verify their previous knowledge regarding the concept of algorithm.
Step A.2: *Modification of student previous knowledge*. In this step, the students are provided with four somewhat similar definitions of the concept of algorithm and they are asked to select the most appropriate among them. Immediate feedback is provided by the system.

Part B: A real life procedure represented in algorithmic manner
Step B.1: *Investigation of student algorithmic approaches*. In this part, the student are asked to describe, in terms of written expressions in natural language, the procedure of preparation of a cup of greek-coffee as a set of simple, necessary and concrete steps. Each expression has to describe only one action. The system provides 13 fields for the editing of 13 expressions as at least 13 simple actions are needed for the preparation of a cup of greek-coffee. By describing the previously mentioned activity students have the opportunity to express their own algorithm for the said procedure. By reflecting on students' descriptions, teacher can plan appropriate interventions. In the next step of this experiment, students can reflect on the feedback given by the system to verify their description.
Step B.2: *Modification of students' algorithmic approaches*. Here, a number of specific expressions are provided for the students to select among these expressions the appropriate ones to describe the procedure of preparation of a cup of greek-coffee. These expressions are in the form of first person, plural (we…). Each of these expressions describes a concrete simple action needed for the above procedure. The system is able to verify the correct selection of each expression and the correctness of the sequence of the expressions selected.

Part C: Commanding the computer to perform a real life procedure using natural language
Step C.1: *Investigation of student approaches*. Here, students are asked to command the computer to perform the real life activity in focus. In fact, students are asked to translate in imperative and in second person of singlular (you…), the specific expressions they worked out during the previous step. These expressions have to be posed in specific text boxes using 'drag and drop'. These boxes are connected by arrows indicating the flow of the actions needed for the said procedure. In fact, the whole picture looks like a primitive flowchart. The teacher can study the flowcharts created by the students in order to plan appropriate interventions.
Step C.2: *Modification of student approaches*.Alternately, students have the chance to submit their flowcharts to the system and receive immediate feedback so that they can modify their actions.

Part D: Real life procedures in the form of primitive programs in natural language
Step D.1: *Students' individual approaches*. Here, students are asked to exploit their previous experience within this environment to form a recipe in a specific commanding form using the expressions they worked out in the previous step (Step C.2). Actually, students are instructed to: a) give their recipes titles, b) define the ingredients required and put them in a separate part of this recipe, c) form the main procedure of preparation of a cup of greek-coffee and put this in a separate section and d) mark out the initialization and the end of the procedure at hand.
Step D.2: *Modifying students approaches*. Here, the software tool provides the student with a number of different fields to write his/her own recipe. In fact, the software provides five fields; starting-field, title-field, ingredients-field, preparation-field, and ending-field. The system provides feedback to the students to modify their attempts.

Part E: Real life procedures in the form of primitive programs in Pseudocode
Transforming student approaches into Pseudocode. Here, the software tool provides students with a skeleton of a recipe in pseudo-Pascal. Students are asked to associate the fields of this recipe with the fields of the recipes they produced in the previous step. In this way, students can be helped to translate the procedure written in natural language into pseudocode. This translation is essential because students have the opportunity to move from human-oriented descriptions to computer-oriented pseudocode-based descriptions. Students can exploit the knowledge they acquired from this translation experience to advance to a stage where they will be able to command the computer to perform a procedure using a specific programming language.

Part F: Forming primitive programs in Pseudocode using basic algorithmic structures
Step F.1: *Exploring dynamic interlinked representations of a program in Pseudocode, using basic algorithmic structures*. In this step, students are expected to have acquired some basic knowledge about algorithmic logic and pseudocode. So, they are provided with opportunities to gain some knowledge about basic algorithmic structures such as; assignment statement, conditionals and iteration structures. To help students to acquire this knowledge, they are encouraged to experiment with interlinked multiple representation systems. These systems consist of a visual

simulation demonstrating the procedure of preparation of a number of cups of greek-coffee and this (simulation) is interlinked with the visual representation of the execution of the program that describes the procedure at hand in pseudocode. To help students acquire deeper knowledge about each individual algorithmic structure embedded in this program, the system provides additional opportunities. In fact, each specific structure is hyperlinked with visual, exploratory and interactive constructions. These constructions can be used by the students for additional experimentation and for verification of their knowledge. The provided interactive constructions are presented in the following section.

Step F.2: *Experimenting with dynamic interlinked representations of the assignment statement*. By clicking on the assignment statement of the program mentioned earlier, students are transferred in a visual metaphor representing this statement. To experiment with this visual metaphor, students can form a specific simple assingment statement eg. X: =X+5. This statement is displayed on the screen of the computer. Visualization of said statement includes 2 boxes representing the left and the right value of X-variable. Students can select values for X. These values are represented as arrows in the left part of the screen. Students can play with these arrows by dragging them and dropping them into a basket and then observing them slowly drop through this basket into the box on the right hand side of the assignment statement. After this experimentation, the student has the chance to assess his/her knowledge by trying a specifically designed quiz. The students' attempts can be checked by the system.

Step F.3: *Experimenting with dynamic interlinked representations of the iteration structures*. Students are provided with a visual interactive simulation of the preparation of a number of cups of greek-coffee to further study the 'for…next' algorithmic structure. In particular, students can control a simulation demonstrating a procedure where a coffee-pot is filled with a number of cups of water and a number of spoons of sugar and greek-coffee. This interactive simulation is interlinked with two types of algorithmic structures representing the previous mentioned procedure: a serial structure and a 'for…next' structure. Students have the chance to control step by step the execution of this simulation and at the same time observe the dynamic execution of these two types of structures. Students have the chance to see these structures in natural language and in pseudo-Pascal.

Step F.4: *Experimenting with conditionals through dynamic animations*. Here the focus is on the control statement. The system provides the students with a simulation of the cooking procedure of the greek-coffee. When the coffee is swells into the coffee-pot, it is ready to serve. Then, the coffee-pot has to be withdrawn from the hotplate and the hotplate must be turned off. This simulation is interlinked with the corresponded expression in pseudocode using natural language and pseudo-Pascal. Here, the control statement indicates that if the coffee is blowing up that means that the hotplate must be turned off.

Technical characteristics of the software. The Macromedia Toolbook II was used.

The pilot evaluation study of SAS

The pilot evaluation study of SAS was realized under the theoretical framework of constructivism regarding the knowledge construction (von Glasersfeld, 1987). This framework emphasizes the students' development during their interaction within a learning environment, taking into account their starting points regarding not only their learning results but also the learning concepts in focus. From a methodological point of view, this evaluation study is a qualitative research (Cohen and Manion, 1989) investigating: a) students' previous knowledge regarding the concepts in focus, b) students' development during their interaction within SAS and c) usability issues regarding this software. The pilot research study was conducted in a secondary scool (6th Gymnasium) of Patras, Greece. A sample of nine students of 9 grades participated in this study. The duration of the students' experimentation within SAS was about 2 hours and followed their needs. Most students have been taught the concept of algorithm and of basic algorithmic structures in BASIC. In the first phase of this study students were given a test with a view to their previous knowedge about the concept of algorithm and of basic algorithmic structures being investigated. A similar test was also given to them after their experimentation within SAS in order to realize their progress and the impact of the software on their knowledge. The researchers undertook the role of observers in order to record everything that students said and did during this experiment and did not affect their actions. The data resources consisted of the automatically created logfiles including the history of each student actions and the field notes of the researchers.

Students' previous knowledge. In the pre-test, students were asked to express their opinion about what they thought an algorithm to be. Students were also asked to write the appropriate pseudo-code for the procedure of preparation of 2 glasses of orange juice with one spoonful sugar in each of them. As emerged from the analysis of the data collected from this test, all except one student expressed that an algorithm is the code of a program in a specific programming language. Nobody was able to write the appropriate pseudocode for the given task. The majority of students (7 students) used I/O statements in BASIC at the same time ignoring to describe the preparation of the

orange juice procedure. So, nobody used assignment statements, conditionals and iteration structures. In fact, students calculated the quantities of the ingredients required for the recipe mentioned above. In our view, it indicates that these students interpreted this task in a mathematical sense where the emphasis is mainly put on calculations.

Students' interactions within SAS. The majority of the students experienced difficulties in describing the the procedure for the preparation of a cup of greek-coffe in discrete steps and using clear expressions in natural language (Step B.1). Usually, students described the whole procedure ambiguously. However, in the next step of the experiment (Step B.2), these students were helped by the presence of the proposed specifc expressions and by the feedback provided; so that, after some attempts, all of them managed to correctly describe the task at hand. Notably, these studens returned to step B.1 and modified their attemts. In step C.1 the majority of students had problems in using the imperative for the construction of the specific expressions used in the previous step, and also to put them in a correct order as well as to put the control statement in the correct position. In the next step (Step C.2) all students were helped by the feedback given by the system and managed to correctly express this procedure. Based on this experience, all students formed a correct commanding recipe for the computer to prepare a cup of greek-coffee, using the specific expressions they worked out in the previous step. To perform correctly in this step of the experiment (Step D.1), students were also helped by the template provided indicating the necessary parts of this simple program. As a result, all students gave titles to their recipes, put the necessary ingredients in a separate section, described correctly the procedure of the task at hand and put it in a separate paragraph, and also marked the starting and the ending points of the whole procedure. These students' recipes were clear, simple and precise in contrast to the ambiguous recipes they formed in step B.1. Unfortunately, all students expressed their recipes in first person, singular. They were helped to correct their attempts by the feedback provided in step D.2. Next, students moved to explore the corresponded recipe in Pseudo Pascal where they observed the one-to-one matching of the expressions used in natural language with the ones used in pseudo Pascal (Part E). It is worth noting that all students were participating with interest and experienced more than once all the interactive animations provided -in Part F- corresponding to the basic algorithmic structures.

Students' progress. In the post-test, students were asked to write in pseudo Pascal the algorithm that describes the preparation of 5 glasses of chocolate milk using 1 spoonful cocoa and 2 spoonfuls sugar per glass. As emerged from the post-test, all students managed to describe correctly the task at hand using pseudo-code in pseudo-Pascal. Six students used a serial description while 3 students used the iteration structure. All students also succeeded in the quiz used to assess their knowledge regarding the assignment statement.

Usability issues. All students used serial navigation as all steps were necessary to grasp the task at hand. All students also expressed interest through all the steps of the activity.

Concluding remarks and future work

A multi-representational, activity-based educational software for the beginners' learning of the concept of algorithm and of basic algorithmic structures is presented in this paper. The design of this environment was realized taking into account social and constructivist perspectives regarding of the knowledge construction. This environment, namely, Starting with Algorithmic Structures (SAS), provides learners with opportunities to express their existing knowledge and also to modify it by being given immediate feedback on their actions. SAS also supports the step-by-step development of students' knowledge regarding algorithmic logic and basic algorithmic structures by providing them with a number of different representation systems to express themselves. The construction of these representation systems has emerged from previous research regarding beginners' thinking about algorithmic logic and algorithmic structures. So, there are no cognitive gaps between the representation systems provided and students' thinking. In addition, SAS supports learning of algorithmic logic through real-life learning activities. Such activities relieve students from the cognitive load stemming from the typical mathematical activities usually used in school-books to introduce students to algorithmic logic. SAS was pilotically evaluated with real students. Students' knowledge was assessed before and after their interaction within SAS through a pre-test and a post- test, respectively. The analysis of the pre-test showed that students confuse algorithms with programs written in a specific language. In addition, students seemed to emphasize input data and output results and ignored the procedure used to produce results from the data given. It was probably due to the fact that students are more familiar with problems posed in a mathematical sense where it is demanded of them to strive for numerical results. This means that algorithmic logic has to acquire a full status in school curricula as a timeless problem solving strategy. Pseudocode also has to acquire a dominant role in algorithmic logic in secondary schools and not to be overlooked by making students use a specific programming language from the very beginning. The analysis of the data emerged from the post test study shows that this educational software can help students to move from vague and blurred views regarding the concept of algorithm to

more clear, precise and whole opinions such as the description of the realization of an activity in concrete steps using pseudocode and basic algorithmic structures. Our future plans consist of evaluating this software with a larger sample of students and enhancing it by adding more real life activities.

References

Allwood, C. (1986). Novices on the computer: a review of the literature. *International Journal of Man-Machine Studies*, Vol. 25, 633-658.

Cohen, L. & Manion, L. (1989). *Research Methods in Education*. London: Routledge.

Christiaen H. (1998). Novice Programming Errors: Misconceptions or Mispresentations? *SIGCSE Bulletin*, Vol. 20, No. 3,5-7.

diSessa, A. (1987). Phenomenology and the Evolution of Intuition. In C. Janvier (Eds), *Problems of representation in teaching and learning of mathematics* (pp. 83-96). London: Lawrence erlbaum associates.

Du Boulay B. (1986). Some difficulties in Learning to Program. *Journal of Educational Computing Research*, 2 (1), 57 73.

Dyfour-Janvier, B., Bednarz, N. & Belanger, M. (1987). Pedagogical considerations concerning the problem of representation. In C. Janvier (Eds), *Problems of representation in teaching and learning of mathematics* (pp. 109-122). London: Lawrence erlbaum associates.

Freund, S.N. & Roberts, E.S. (1996). THETIS: An ANSI C programming environment designed for introductory use. *ACM, SIGCSE '96* 2/96, Philadelphia, PA USA,300-304.

Jonassen, H. D. (1996). *Computers in the Classroom*. Colombus, OH: Prentice Hall.

Kordaki, M. Miatidis, M. and Kapsampelis, G. (2005). Multi Representation Systems in the Design of a Microworld for the Learning of Sorting Algorithms. In Proceedings of *IADIS International Conference Cognition and Exploratory Learning in Digital Age* (CELDA 2005), Porto, Portugal, 14 -16 December 2005, pp. 363-366.

Kurland, M. (1989). The Study of the development of Programming Ability and Thinking Skills in High School Students. In E.

Soloway & J. Spohrer eds), *Studying the Novice Programmer* (pp.83-112). Hillsdale, NJ: Lawrence Erlbaum associates.

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.

Noss, R., & Hoyles, C., 1996. *Windows on mathematical meanings:Learning Cultures and Computers*. Dordrecht : Kluwer Academic Publishers.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.

Pattis, R.E. , Roberts, J. & Stehlic, M. (1995). *Karel-The Robot , A Gentle Introduction to the Art of Programming*. 2nd ed., New York, Wiley.

Samurcay, R. (1989). The Concept of Variable in Programming: Its Meaning and Use in Problem Solving by Novice Programmers. In E. Soloway & J. Spohrer eds), *Studying the Novice Programmer* (pp.161-178). Hillsdale, NJ: Lawrence Erlbaum associates.

Satratzemi M., Hadjiathanasiou K., Dagdilelis V. (2000). Anim Pascal: An educational tool for the support of introductory Lessons in Programming. 2nd Panhellenic Conference "*The Information and Communication Technologies in Education*", 125-135, Patras, Ed. 'Nees Tehnologies, 2000.

Soloway E. Spohrer J.C. (1989). *Studying the novice programmer*. Hillside, N.J. Erlbaum.

von Glasersfeld, E. (1987). Learning as a constructive activity. In C. Janvier (Eds), *Problems of representation in teaching and learning of mathematics* (pp.3-18). London: Lawrence Erlbaum.

Vygotsky, L. (1978). Mind in Society. Cambridge: Harvard University Press.