Using a computer-based interactive system for the development of basic algorithmic and programming skills

Maria Marcelino, Anabela Gomes, Nickolai Dimitrov, António Mendes

Abstract:. *Student failure and commonly expressed difficulties in programming disciplines suggest that traditional approaches are not the most appropriate for many students. In this paper we present SICAS, a learning tool designed to help students in the development of basic algorithmic and programming skills. With SICAS students can design, simulate, test and compare algorithms for proposed problems. We also report some evaluation findings, current changes and future extensions of the tool.*

Key words: *algorithm animation, software visualization, educational technology, programming teaching and learning.*

INTRODUCTION

It is well known that many students have difficulties in the learning of programming, but this becomes more critical when we are talking about Informatics Engineering students of College level as this is a key competence for them.

The learning of programming requires skills like abstraction, generalization, transfer and critical thinking, among others. These skills are hard to develop and students usually evidence a clear lack of training in this respect. For this development traditional methods and approaches seem ineffective too, because they are based mainly in static materials, which we think are not appropriate for a subject matter that is essentially dynamic [1].

Experience has also shown that the problem starts, in general, at the initial phase of learning – in understanding and applying certain abstract programming concepts, like control structures, in creating algorithms that solve concrete problems, in mastering the language to express algorithms, etc.

Several approaches and tools have appeared with the intention of supporting the programming learning process in different ways. However, though we find reports of positive results from the utilization of some of these tools [2], none of them has gained a generalized utilization.

So, the idea of designing and implementing an educational computer-tool to help the learning of the basic mechanisms of programming oriented to the design and implementation of algorithms came up. We called it SICAS. SICAS doesn't include any expositive materials, on the contrary is an environment that allows students to develop their capacities by experimentation, allowing them to design, observe, analyze and simulate algorithms, making possible for them to detect errors, correct them and learn from them. So it offers and is based on a constructivist approach to learning, where each student learns at his own pace and progressively constructs his own knowledge. We strongly believe, like many researchers, that this kind of approach can improve student problem solving capacities as well as their critical thinking capabilities [3].

In this paper we start by describing the main characteristics of SICAS in its first version, SICAS 1.0. After we present the conclusions of a set of tests we have made with this version of the tool with teachers and students in real settings. We also report the changes that are currently being made under version number 2 and some future related extensions.

SICAS 1.0

SICAS 1.0 allows essentially two types of activities: design/edition of algorithms and execution/simulation of algorithms.

In the first case, the user constructs algorithms using flowcharts with typical graphical representation symbols (see Figure 1 for an example). In the second case, the user can simulate and view in animated form the execution of algorithms.

SICAS has also two modes of working - student mode and teacher mode.

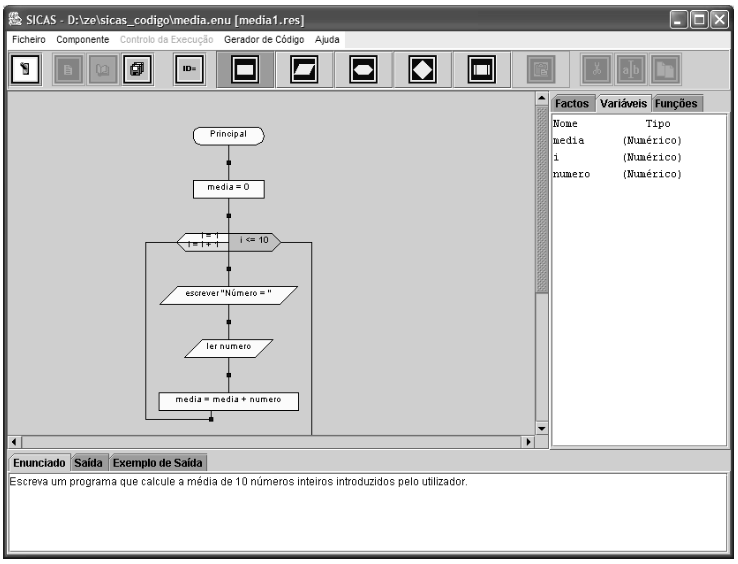
In the teacher mode, a teacher can create a set of problems. A problem consists of a problem formulation, one or more algorithms that solve it and, eventually, some data test sets. The existence of several algorithms for the same problem allows for different reasoning types and forms of understanding, allowing the student to compare them and find out which resolution strategy is more adequate for a given problem. The data test sets allow the student to easily verify if his solution 'really' solves the problem.

Algorithm design is supported by an iconic environment, where the user can build flowcharts to represent them. We used flowcharts, instead of pseudo-code, because many studies reinforce that this form of representation is more appellative, facilitates understanding, is simpler and probably less prone to errors than pseudo-code [4].

The structures that can be used when creating a flowchart in SICAS are:

- Assignment – to set the value of a variable with the result of an expression.
- Input/Output - to read values from the user or to write them in the output window.
- Repetition - to repeat the execution of some action.
- Selection – to choose between two sets of actions that may be executed.

These elements can be introduced in a flowchart by clicking (in the toolbar icons) and pointing (in the design area). They can also be inserted through menu selection. In any case, the dialog boxes that automatically open to specify element details were designed to include the minimum information necessary, in order to avoid common novice programmer syntax errors (see Figure 1). Also lines connecting components are automatically inserted, avoiding inconsistencies in the flowchart. At any moment it is possible to delete, modify or copy any component.



```
Nome          Tipo
media       (Numérico)
i           (Numérico)
numero      (Numérico)
```

Escreva um programa que calcule a média de 10 números inteiros introduzidos pelo utilizador.

Figure 1 – Example of algorithm design.

SICAS only supports variables of the following data types: "Numbers", "Strings", "Arrays of strings" and "Arrays of numbers", but as it intends to support basic programming learning these data types seem sufficient for the moment.

The construction of expressions in SICAS uses a syntax similar to that of C and JAVA, because the environment potential users will program, at a later stage, in one of these languages. However, we minimized syntactic details, so that the student concentrates completely in the creation of algorithms.

Another important feature of the tool is the possibility for students to define functions. This tries to introduce them to the concept of modularization, which has several advantages (algorithm legibility, complexity management, re-use of components, etc.).

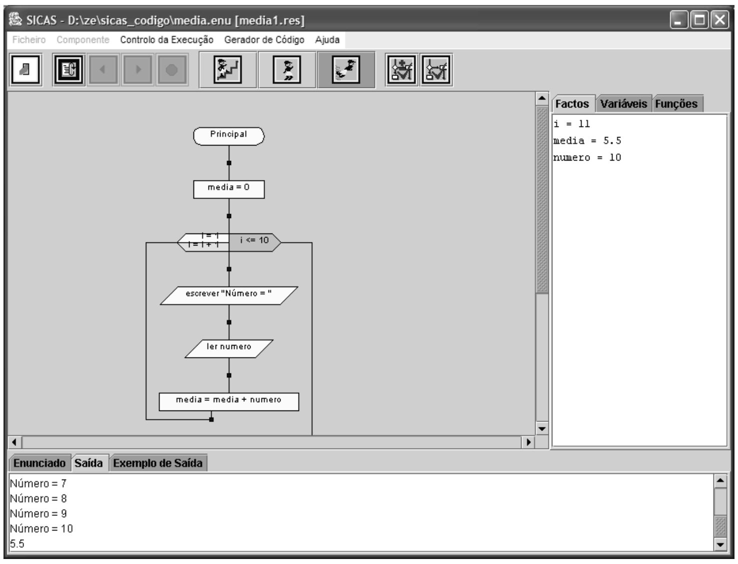
Function construction is done using the mentioned structures and according to the rules and options previously described. SICAS also have a set of pre-defined functions that can be used in expressions (for number and string manipulation).

Any algorithm created with SICAS can be automatically translated to pseudo-code, C or JAVA code. These various alternatives show to students that a well designed algorithm can be easily translated to several programming languages and that the most important factor in algorithm design is its conception, not the programming language in which it will be coded.

After building an algorithm, the student can see its animated simulation. The student can:

- Control the rhythm at which the simulation progresses (step-by-step, slow or fast).
- Pause the simulation, allowing a deeper analysis of available data and/or a discussion with the teacher or other learners.
- Go back and repeat some part of the execution.

During the simulation the component in execution is highlighted with a different color. Figure 2 shows an aspect of SICAS during an algorithm simulation.



```
i = 11
media = 5.5
numero = 10
```

```
Número = 7
Número = 8
Número = 9
Número = 10
5.5
```

Figure 2 – Example of algorithm simulation.

If the teacher has provided a set of test data for the problem (inputs and outputs), the student can test his solution with those data, hence validating it. This is particularly useful for special cases (e.g. invalid data or data that will likely produce erroneous situations), since many students tend to get satisfied if their solutions work for the average case, without caring to see if they work in any case.

SICAS 1.0 EVALUATION

SICAS 1.0 has been submitted to several evaluation tests. First informal evaluations with programming teachers were carried out. Later, tests with students in classroom scenarios took place.

In the first case, the general opinion was that SICAS is an interesting and useful product for introductory programming disciplines, to be used either in the classroom or to support teacher and student autonomous activities. They considered the graphical interface agreeable and intuitive. Most teachers did not have the necessity of reading the manual before using it, for instances. They liked specially the possibility of making a function call in a flowchart, since it allows introducing the topic at the beginning of the learning, which is usually not the case. So, in this way, SICAS helps to overcome this gap.

Most teachers thought that it was also important to call the student attention to data types (numeric, alphanumeric, arrays), which is not also a current practice in algorithm design and that they think it is pertinent. Usually this topic is left to when one starts programming, which is not a very correct strategy for some of them. On the other hand, having only a few data types is good, at an initial learning phase, since it does not overload the student, but it maximizes his concentration in the essential - the design strategies to solve a problem.

Above all, they liked algorithm simulation, seeing this as the strongest aspect of SICAS. For them, many times, it is quite dull to pick up a piece of paper and a pen (or a piece of chalk and a blackboard) and simulate an algorithm. Also it is time-consuming. So, most teachers thought that that possibility was very interesting, mainly step-by-step simulation. They also appreciate the possibility of expected result validation, mainly because it contributes to logical error detection.

Relatively to the second group mentioned, evaluation with students, it was done in two ways. First SICAS was distributed to all the students enrolled in the disciplines of "Programação e Algoritmos I" of the Informatics Engineering Degree of the Department of Informatics Engineering of the University of Coimbra and of "Programação I" of the Informatics Engineering and Systems Degree of the Department of Informatics Engineering and Systems of the Polytechnic Institute of Coimbra. These are both introductory programming disciplines of the first year on both degrees. The system was accessible through a server along with the user manual, examples of algorithms and their simulations. The second evaluation was done only in "Programação e Algoritmos I" classrooms. This experience had as evaluation instruments direct observation, task analysis and comment and opinion collection.

In the discipline classes, each teacher started by presenting SICAS very briefly, pointing out main functionalities, characteristics, activities and the meaning of the several icons. Afterwards, students were organized in groups of two elements each. Groups were generally heterogeneous, with novice students to programming, mixed with students that already are used to program, in one or more programming languages. Each group was proposed to do the following activity: they had to translate to flowchart a solution written in pseudo-code in the blackboard, by the teacher. The idea was to evaluate essentially the facility in moving around in SICAS. Each teacher tried to accompany the groups as much as possible, so that he could register student reactions and the types of difficulties expressed and felt pertinent, namely:

- activity completion mean time per group;
- the number of times that the group made a mistake in the type of element that they wanted to insert in the flowchart and why;
- things that they should have done and didn't because they forgot to do or because they did not know that it was necessary to do;
- the number and type of help that was asked and given;
- the type of errors made in the insertion of objects, etc.

Due to the difficulty for a teacher in accompanying all the class groups, students were encouraged to use the field comment in each component, to register eventual doubts/critics.

In the following lecture, each student received paper sheets with print screens of both SICAS screens (design/edition of an algorithm and execution/simulation of an algorithm). The objective was to determine what each icon suggested to each student, measuring in this way the intuitive nature of the interface. Initially we thought to do this activity in two ways: one where the icons were isolated and another where they were grouped as in SICAS. After we opted only by this last one because, we think that is more correct as it is the context in which they are going to be used. We concluded the following:

Although most students could identify the fundamental icons of the design/edition module of the system, there were some students that were not able to identify any icon. Some identified the icon to change between design/edition to execution/simulation as testing (when in the execution/simulation module) and to coming back to the problem (when in the design/edition module).

After SICAS was used in the classroom substituting traditional media, paper and pen, to build algorithms. During the first utilization of SICAS, students used mainly the menus to select the operations to do, instead of the icons – the icons were not felt very suggestive. We think that this happened because the majority of the students used pseudo-code to express algorithms and only a minority used flowcharts, although a few used both alternatively. Even in "Programação e Algoritmos I" classes pseudo-code is traditionally used. But after a period of SICAS utilization, students left the menus and start using the icons.

SICAS 2.0

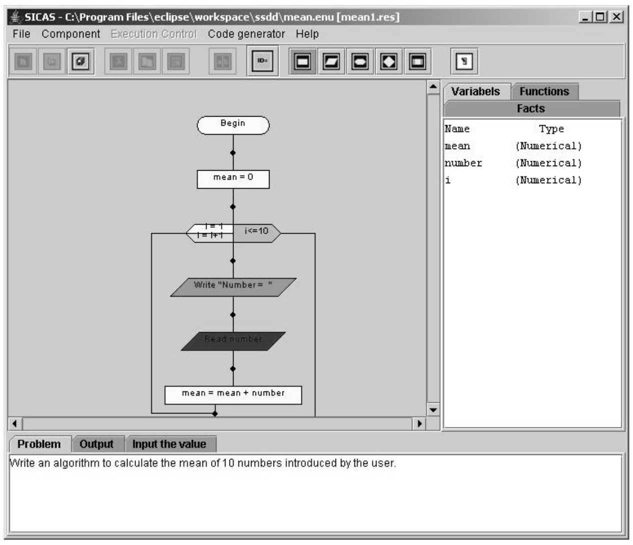
From the analysis of student reactions during their task execution in classes, from comments received from the students that used SICAS extra-classes to support their study and from other evaluator comments we decided to implement a new version of SICAS. In this new version improvements were considered mainly at two levels, in terms of a more intuitive interface that ease browsing and in terms of other functionalities that revealed important.

First small changes were made like reducing the size of the icons and enlarging the size of the design window. Other changes included the refining of certain error messages, some extra edition capabilities and the icon re-organization in the toolbars.

Major changes included the possibility of specifying an algorithm in pseudo-code and the interface language adaptability.

SICAS 1.0 generated the pseudo-code corresponding to a flowchart, but not the opposite. In this new version we wanted to implement this feature. So, now it is possible to express an algorithm in flowchart or in pseudo-code format and generate the other.

One clear barrier to the utilization and spread of a software package is the interface language it uses. This is the case for most products in the Portuguese market. If they are not in Portuguese they will not be used by most students. So the new version of SICAS is designed in a way adaptable to the user language. In Figure 3 we can see a screen of the product in English, but it can be adapted easily to any other language.



Name	Type
mean	(Numerical)
number	(Numerical)
i	(Numerical)

Write an algorithm to calculate the mean of 10 numbers introduced by the user.

Figure 3 –English version of SICAS.

This same idea can be applied to pseudo-code. Although pseudo-code uses a simpler language than our every day language, the words it is based on should be in the user mother language. For that purpose SICAS 2.0 also allows to choose and adapt the pseudo-code language (Portuguese, English, etc.).

CONCLUSIONS AND RELATED WORK

SICAS is an educational tool that aims at helping student to understand and master the utilization of basic programming structures in a constructivist perspective. The student can design algorithms that solve concrete programming problems proposed by a teacher or by his own initiative. The system can be used by students alone or in-group, in classes or in a self-study format. This can allow weaker students to improve their programming knowledge and experience.

From the study done we are encouraged to, in the next curricular year, use only SICAS 2.0 to support algorithm development with our students and abandon the paper and pen method. We expect to gain in student productivity and global understanding as well as in the discipline rate of success.

Learning theories point out that students learn better if not isolated. So, another version of SICAS, a collaborative one, is being developed that will allow the shared construction of algorithms by students even situated in geographically different places to take place easily. This new version will also support the discussion of the proposed solution, among other things [5].

REFERENCES

[1] Gomes, A. and Mendes, A. J., "A animação na aprendizagem de conceitos básicos de programação", *Revista de Enseñanza y Tecnología*, Nº 13, pp. 22-32, 1999.

[2] Lawrence, A., Badre, A. and Stasko, J., "Empirically Evaluating the Use of Animations to Teach Algorithms" *1994 IEEE Symposium on Visual Languages*. St. Louis, pp. 48-54, 1994.

[3] Ben-Ari, M., "Constructivism in Computer Science Education", *Journal of Computers in Mathematics & Science Teaching*, 20 (1), pp. 45-73, 2001.

[4] Scanlan, D., "Structured Flowcharts Outperform Pseudocode: An Experimental Comparison", *IEEE Software*, Vol. 6, Nº 5, pp. 28-36, 1989.

[5] Redondo, M., Bravo, C., Marcelino, M. and Mendes, A.J., "Tools for programming learning: an approach to provide a social perspective using collaborative planning of design", *IADIS International Conference e-Society 2004*, Avila, Spain, July 2004 (accepted for presentation).

ABOUT THE AUTHORS

Assist.Prof. Maria Marcelino, PhD, Department of Informatics Engineering, University of Coimbra, Phone: +351 239 790000, E-mail: zemar@dei.uc.pt.

Adj.Prof. Anabela Gomes, MSc, Department of Informatics Engineering and Systems, Polytechnic Institute of Coimbra, Phone: +351 239 790 350, E-mail: anabela@isec.pt.

Nickolai Dimitrov, BSc, Department of Computer Systems, University of Rousse, E-mail: n_ganchev@abv.bg.

Assist.Prof. António Mendes, PhD, Department of Informatics Engineering, University of Coimbra, Phone: +351 239 790000, E-mail: toze@dei.uc.pt.