Algorithmi: Bridging the Algorithms to Natural and Programming Languages

António Manso
Departmental Unit of Information Technologies
Polytechnic Institute of Tomar
Tomar, Portugal
manso@ipt.pt

Célio Gonçalo Marques
Departmental Information Technology Unit
Polytechnic Institute of Tomar
Tomar, Portugal
celiomarques@ipt.pt

Paulo Santos
Departmental Unit of Information Technologies
Polytechnic Institute of Tomar
Tomar, Portugal

Luís Lopes
Student
Polytechnic Institute of Tomar
Tomar, Portugal
aluno19055@ ipt.pt

Raquel Guedes
Student
Polytechnic Institute of Tom
Tomar, Port ugal
aluno19994@ ipt.pt

Computer programming is a crucial competency in computer science and is learned through languages and IDE services designed to develop software in an industrial context. Invariably, these environments and programming languages use the English lexicon, and this can be an obstacle especially for users who do not a good command of the language. Algorithmi is a system that allows the learning of algorithms using the native language of the students in its specification and allows automatic translation into the most popular programming languages. This paper presents the language Generic Algorithm Language (GAL) and its translation into Portuguese, English and Chinese. In this paper we also show the translation of GAL for the procedural programming models (C and Python), the Object-Oriented model (Java, C # and C ++) and the Web programming model through the PHP (server-side) and Javascript language (Client-side). In addition to allow algorithms to be programmed, Algorithmi can also be executed inside the IDE or translated into HTML, CSS and Javascript and executed in the browser.

Keywords - Algorithmi, Learning, Information System, Programming.

I. Introduction

Computer programming is a core competency that is crucial to several fields of computer science and lack of knowledge in this area undermines learning of other content that builds on it.

Learning to program is not easy because it requires the combination of personal and cognitive skills that students need to develop. The absence of these competencies leads to a high failure rate in introductory programming modules which makes it difficult for students to progress and complete their degrees [1] [2] [3] [4]. Programming means thinking abstractly and requires a lot of motivation and commitment. For this reason, we have seen the replacement of traditional teaching approaches with more motivating technological solutions, such as Logo [5] or Scratch [6].

These tools are extremely fun and is easy to learn programming with it, but they are very different from the commercial programming languages that students will find in the job market. The programming environments of these languages have a friendly interface, often in the mother tongue of the students. However, the algorithms do not have a practical application because the problems that serve for learning have a more playful context. One of the most important programming tasks is the automatic mathematical calculation through computational expressions. If languages do not support these calculations, then its applicability boils down to the teaching of computational logic.

In order to provide a tool for teaching and learning computational algorithms that smooths the transition to traditional programming languages and environments we created Algorithmi. Algorithmi is an evolution of Portugol IDE [7], [8], and provides a learning environment for computational algorithms in which students use the language that is based on their native language. The editing and execution of the algorithm is based on flowcharts, and the tool allows its translation into the most popular programming languages. This facility enables students to view the algorithms in their native language and their translation into various programming languages and paradigms. This paper describes how Algorithmi translates the algorithm into the natural languages of the students and programming languages used in the software development industry.

II. Natural Languages and Programming Languages

Idioms and languages, although related, are different concepts. Idiom is an organized set of elements that enables communication. On the other hand, language is the ability to produce, develop and understand an idiom. We can consider idiom as the tool used by a language to materialize.

A. Natural Languages

Most existing languages are natural languages that were born from oral communication between individuals and have later evolved to written form and symbolic representation.

When writing was created, human beings attempted to give a visual representation to the words they used in their oral communication and this representation in the early writings was essentially ideographic, i.e. each symbol represented an idea. It is curious how civilizations with no apparent contact like the Egyptians or the Mayans, both developed an ideographic writing based on hieroglyphs. However, it was the Phoenicians who created a writing system based on phonetic sounds associating a symbol with each phonetic sound [9] which eventually gave rise to the Greek, Latin and Arabic alphabet that is used by most languages. The Chinese writing system, which gave rise to Japanese and Korean (also widely used) incorporates features of ideographic writing and

phonetic writing, i.e. symbols representing ideas and symbols representing sounds.

Language evolution has been exponential, slow and time-consuming at first. Oral language took thousands of years to evolve from small sounds to a vocabulary that allowed rich and efficient communication between individuals and evolved more rapidly as people were getting to know each other and creating cultural and commercial relations. Or was simply imposed by military power through territorial conquests.

The Portuguese language, for example, derives from Latin, imposed by the military expansion of the Roman Empire, with influences of the Celtic spoken by the Iberian peoples that previously inhabited the region, with Germanic influences imposed by the Suevi and the Visigoths that followed the Romans and with influence of the Arabic through the Muslim invasion of the Iberian Peninsula. All these events influenced the Portuguese language only in hundreds of years. Today, languages are influenced not in thousands of years, not hundreds, but only in tens. The increase in schooling, the removal of borders imposed by common economic areas, emigration and globalization have put pressure on languages.

All these pressures make languages evolve, often faster than their safeguarding bodies would desire.
Each language has its own rules but in day-to-day communication people often infringe some of these rules and this forces it to evolve.

B. Computational languages

Although computational languages seem to different from natural languages, their origins are very similar and they serve the same purpose: to communicate a message. Natural language arises from the need of individuals to communicate with each other, just as computational languages arise from people's need to communicate with computers.

So, if natural languages are subject to pressures that lead to their evolution, a similar thing happens with computational languages. Just as natural languages have their grammatical rules, computational languages have their syntactic rules. However, it is in these rules that lies the main difference between these two types of language.

In natural language, a severe infringement of grammatical rules may prevent the other party from understanding the message, but a slight mistake may not prevent the message from getting through. But in computational languages even a slight infringement of the syntactic rules is enough to prevent the computer from perceiving the message.

Computers use the 0/1 alphabet to represent messages that are transmitted to them by humans using natural language. Computational languages have evolved over the last 70 years starting [10] with those of the 1st Generation (where Machine Languages fits in), in which the programmer had to have a deep knowledge about the computer to communicate with it. Those of the 2nd Generation, which include Assembler. Those of the 3rd Generation, those of high level, where the procedural languages fit with Basic, Pascal, C, Cobol or Fortran. Those of the 4th Generation, which were highly specialized languages in some tasks, such as DBASE, SQL or Clipper. To conclude, those of the 5th Generation that include object-oriented languages such as C++, Java, functional languages such as LISP and logical languages such as PROLOG.

All computer languages have in common the fact that there is then a translation process that translates the message written on that language into machine language so that the computer can understand it. It is in this process of translation that a slight breach of the syntactic rules is enough to prevent the computer from decoding it.

Comparing the evolution of the writing of natural languages with the writing of computational languages, the first began with ideographic symbols and later evolved into a phonetic representation, the second ones always tried to give the programmer a more natural way of talking to the computer, but never evolved beyond that. They evolved in the context of solving and addressing the problems to be solved, with the creation of several programming paradigms, but in their essence, these paradigms are encoded with a set of similar instructions. When learning programming, it is fundamental for students to master the language of interaction with the computer, and the computational languages seem to have all been designed to solve complex problems and not for teaching programming.

III. ALGORITHMIC LANGUAGE

The algorithmic language is an evolution of Portugol, whose main characteristic is the fact that the students write the algorithms in the Portuguese language. The language was designed to teach programming and contains only what is essential for this purpose. This simplification of general-purpose computational languages leads to a reduced lexicon. Algorithmi is a programming learning environment that uses a formal language based on tokens, which is used to represent and execute algorithms – the so-called General Algorithm Language or *GAL* for short.

GAL is the language used by the core to verify and execute algorithms in the learning system, Edition and execution of algorithms is done by translating reserved words – tokens – to corresponding words in the student's mother tongue.

The GAL language predefined the keyboard as input device and the screen as output device. The algorithms are represented by computational instructions that manipulate variables, to which a data type is associated (Table 1) by reading the keyboard or by processing through operators and functions (Table 2). The result of processing is displayed in the console. The delimiters and operators are static and defined in the core of the language, however everything else is dynamic and can be translated into natural language words.

Table 1 presents the GAL simple data types in Portuguese, English and Chinese. The void type was introduced to allow for the setting procedures, namely functions that do not perform calculations. The integer and real types allow you to represent numeric data, the logical type represents the true and false values, and the text type allows you to represent a set of characters.

TABLE 1 – TYPES OF DATA

GAL	**Portuguese**	**Chinese**	**English**
VOID	vazio	空虚	void
INTEGER	inteiro	整数	integer
REAL	real	真实	real

TEXT	texto	串	text
LOGIC	lógico	布尔	logic

TABLE 2 – FUNCTIONS

GAL	Portuguese	Chinese	English
INT	int	整数	int
ELEMENTSOF	length	要点	elementsof
SIN	sin	罪	sin
COS	cos	余弦	cos
TAN	tan	黄褐色	tan
ASIN	asin	反正弦	asin
ACOS	acos	反余弦	acos
ATAN	atan	反正切	atan
SINH	sinh	双曲正弦	sinh
COSH	cosh	双值的双曲余弦值	cosh
TANH	tanh	正切	tanh
ABS	abs	绝对	abs
EXP	exp	指数	exp
INT	int	整数	int
LOG	log	日志	log
LN	ln	自然对数	ln
MIN	min	分	min
MAX	max	最大	max
POW	pow	提升到了权力	pow
SQRT	sqrt	开方	sqrt
RANDOM	random	随机	random

The functions defined in Table 2 are the ones that we consider necessary to develop computational algorithms and can be translated into the most popular computer languages.

TABLE 3 – CONTROL STRUCTURES

GAL	Portuguese	Chinese	English
BEGIN	inicio	开始	begin
END	fim	结束	end
DEFINE	definir	限定	define
READ	ler	读	read
EXECUTE	executar	执行	execute
WRITE	escrever	写	write
IF ELSE	se senão	如果其他	if else
WHILE	enquanto	而	while
DO	faz	做	do
ITERATE	iterar	对于	for
JUMP	saltar	打破	jump
FUNCTION	função	功能	function
RETURN	retornar	返回	return

Table 3 presents the instructions that are defined in the language. This set of instructions is what is strictly necessary to teach and learn the basics of algorithm for problem solving. All algorithms start with *BEGIN* and end with *END* this also allows closing blocks of instructions. The *DEFINE* statement allows the definition and initialization of variables and *EXECUTE* allows you to change its contents through computational expressions. The interaction with the console is done through *READ* for the user to enter data and *WRITE* to write information to the console. The *IF ELSE* statement allows you to make decisions with conditional expressions, and the *DO*, *WHILE*, and *ITERATE* statements allow to repeat a block of statements. The *JUMP* instruction allows you to change the instruction flow within a cycle in conditional shape, allowing you to break or continue the cycle. The *FUNCTION* instruction allows the user to define functions by allowing the modularization of algorithms and the *RETURN* instruction allows the return of a value from functions that perform calculations. The *EXECUTE* statement also allows the execution of procedures (Functions that do not calculate values).

Alg. 1 shows an algorithm, written in GAL language, that prompts the user for their name and then prints a greeting. The program exemplifies some of the characteristics of the language, namely the definition and reading of variables, and writing on the screen of text and computational expressions.

ALG. 1- PROGRAM THAT PRESENTS A COMPLIANCE WITH USER IN GAL LANGUAGE.

```
BEGIN MAIN_PROGRAM_NAME
    DEFINE TEXT txt SET "Hello "
    WRITE "What's your name? "
    READ TEXT name
    WRITE txt SUM name SUM "!"
```

```
END MAIN_PROGRAM_NAME
```

In the following sections we present the algorithm approaching the translation of the algorithm for natural languages and computational languages.

A. Translation for natural languages

Algorithmi allows to translate the algorithm into any natural language as long as the equivalences between GAL and its words in the native language are predefined, as shown in Table 2 and Table 3. The translation of the algorithm for the natural languages is done translating the tokens for the words into the desired language and Alg. 2 presents the translation of Alg. 1 to the Portuguese, English and Chinese languages.

ALG. 2 - TRANSLATION FROM ALG. 1 TO PORTUGUESE, ENGLISH AND CHINESE LANGUAGES.

```
inicio Programa principal
     definir texto txt = "Hello "
     escrever "What's your name? "
     ler texto name
     escrever txt + name + "!"
fim Programa principal
```

```
begin MainProgram
     define string txt = "Hello "
     write "What's your name? "
     read texto name
     write txt + name + "!"
end MainProgram
```

```
开始 主程序
     限定 串 txt = "Hello "
     写 "What's your name ? "
     读 串 name
     写 txt + name + "!"
结束 主程序
```

The algorithm natively has some languages but it allows the new language to be added. To create translations, a small application was developed, Figure 1, which allows to add new languages in a simple and practical way.

As shown in Figure 1, there are 3 different panels. The first panel has a listing of the GAL language keys that are to be translated. The second has the translation of the keys in the languages that are available in the application. The user can change the translation language whenever he wishes so that he can clarify any doubts that the current translation presents and the third one is used to make the translation.

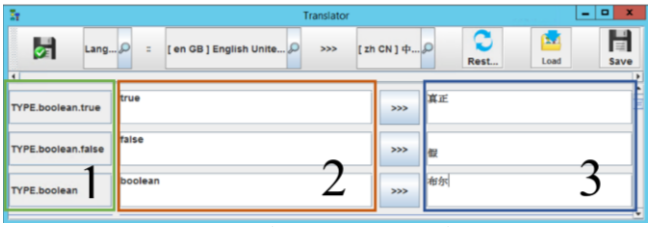

Figure 1 – Translator of natural languages of the application

Translations are stored in XML files that can be edited independently by more experienced users. These files are automatically read by the system in the application language folder without any application compilation required.

The algorithms represented above are translated automatically using available languages of the list represented in Figure 2.

The available languages on the list are Extensible Markup Language (XML) files produced by the application represented in Figure 1.

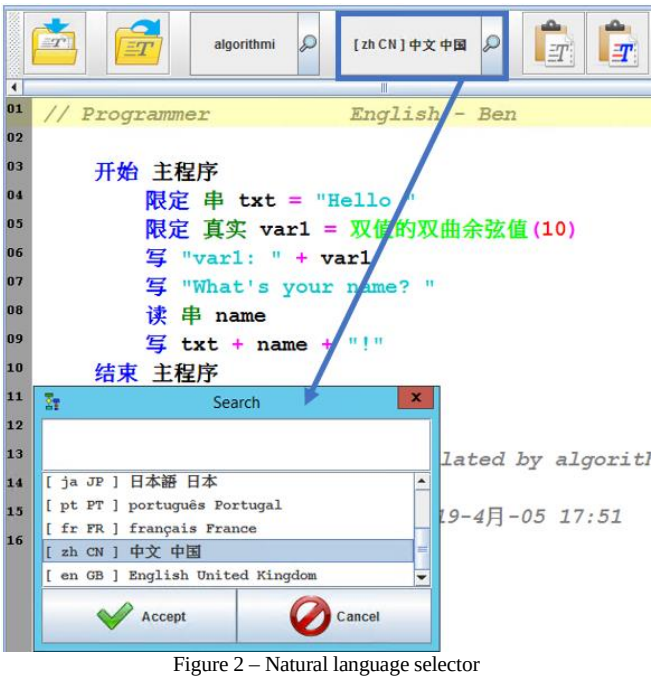

Figure 2 – Natural language selector

Algorithmi allows to edit and visualize the algorithms in flowchart form. The flowcharts replace the GAL instructions of Table 3 by graphic shapes, where computational expressions will be written.

Alg. 1 has a graphical representation in flowchart form (Figure 3, Figure 4 e Figure 5). In the flowcharts, the instructions in Table 3 are translated by the flowchart's symbols, the words that are present in Table 1 and Table 2 are translated into the language of the user. In computational expressions, the names of variables, numerical constants and texts are kept such as, for example, the constant "Hello".

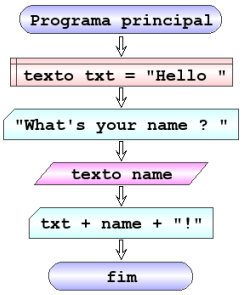

Figure 3 – Flowchart algorithm in Portuguese

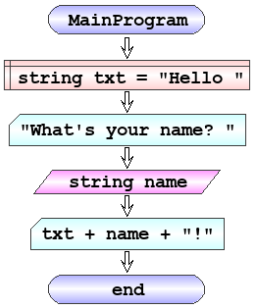

Figure 4 – Flowchart algorithm in English

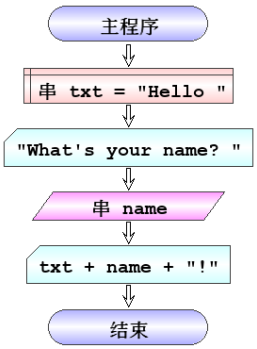

Figure 5 – Flowchart algorithm in Chinese

B. *Translation for programming languages*

Translation for programming languages is much more complex. Natural languages have a direct translation because the GAL language was developed for this purpose, in addition to being interpreted by humans who use their intelligence to make sense of words.

Programming languages are formal languages that are compiled and executed by computers. The rules for writing programs are strict and vary from language to language. Programming languages invariably use English words in their lexicon and differ from one another in the syntax and semantics of computational instructions.

As shown in Figure 6, the users can change easily from one language to another. This way of visualizing translations for programming languages facilitate student learning because they can start with basic programs and increase difficulty in order understand how the language works, using always their native language to make the programs.

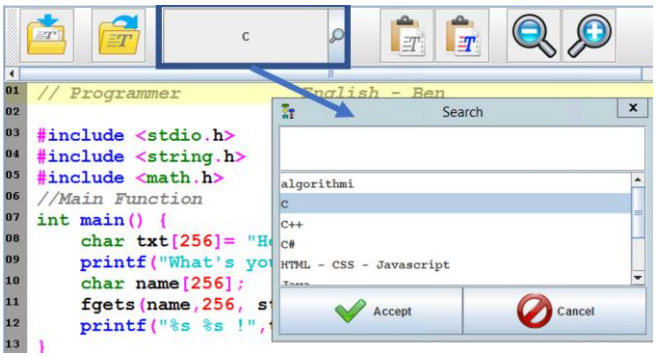

Figure 6 – Programing language selector

1) *Procedural languages*

Procedural languages are high-level languages that use an ordered set of instructions to write programs. The procedural name comes from the fact that there are sets of instructions grouped into functions or procedures. This feature allows you to take a top-down approach to solving complex problems by dividing them into simpler problems. This approach to problem solving is one of the basic skills in computer programming and most programming languages use the concept of function or variants of it.

Algorithmi allows you to automatically translate to procedural languages both C and Python.

C language is one of the most popular general-purpose languages that provides compilers for a range of computer system architectures. It was developed in the early 1970s by Ken Thompson and Dennis Ritchie, it is a standardized language (ISO) and gave rise to some of the more modern languages such as C ++, which share much of its syntax and semantics.

The executable programs generated from this language are very efficient in the use of computational resources which makes it the preferred language for programming critical systems such as operating systems like Unix, for example. Recently with the appearance of the Internet of Things, IOT, which uses devices with reduced computational power, the C language is very used in programming.

The Python language is a modern programming language that supports several programming paradigms. Launched by Guido van Rossum in 1991, it is currently maintained by a community of programmers who develop specialized libraries that allow them to expand language capabilities. It is a language with increasing popularity and its syntax and semantics is based on a set of principles known as The Zen of Python[11] that promotes its simplicity and beauty.

ALG. 3- ALGORITHM TRANSLATION REPRESENTED ALG. 1 FOR C AND PYTHON LANGUAGES.

```c
#include <stdio.h>
int main() {
    char txt[256]= "Hello ";
    printf("What's your name ? ");
    char name[256];
    fgets(name,256, stdin);
    printf("%s %s !",txt,name);
}
```

```python
txt = "Hello "
print( "What's your name ? " , end="")
name =  str(input())
print(txt, name,  "!" , end="")
```

Alg. 3 presents the translation Alg. 1 for C and Python language. This example demonstrates how different languages can be, and therefore there is no common procedure for translating algorithms into programming languages.

In Alg. 3, C language requires the definition of variables with data type, whereas the Python language does not, . C language does not have the native type string, therefore a vector of 256 characters must be defined whereas in Python the string is a native dynamic object.

C language has the braces ("{" and "}") to delimit blocks whereas Python uses indentation. These features are visible in a very simple program (Alg. 3), the differences between the languages are so large that it is not possible to define a generic procedure to do their translation. This translation will have to be done through specialized code, made by experienced programmers in the language in order to be able to obtain programs that can be compiled and executed by the computer.

2) Object oriented Languages

Object-oriented languages allow the programming of applications based on composition and iteration between objects. These objects are the fundamental unit of problem solving that are defined through models given the name of classes. This is one of the most successful programming models in systems programming and there are many languages that support it.

ALG. 4 - ALGORITHMO TRANSLATION REPRESENTED ALG. 1 FOR THE LANGUAGES JAVA, C# AND C++.

```
import java.util.Scanner;
public class Hello {
   static Scanner keyb = new Scanner(System.in);

public static void main( String[] args) {
  String txt = "Hello ";
 System.out.print("What's your name?");
 String name = keyb.nextLine();
  System.out.print(txt + name + "!");
}
}
```

```
using System;
public class Hello {
   static void Main() {
   String txt = "Hello ";
   Console.Write("What's your name?");
   String name = Console.ReadLine();
   Console.Write(txt + name + "!");
 }
}
```

```
#include <iostream>
using namespace std;
class Hello {
    public:
      int main() {
        string txt  = "Hello ";
        cout << "What's your name ? ";
        string name;
        cin >> name;
        cout << txt << name << "!";
     }
};
int main(){
    Hello application;
    application.main();
}
```

Algorithmi allows translation of algorithms for Java, C # and C ++ languages that are shown in Alg. 4.

All these languages share the same programing model and use "Class" as the reserved word to define their object model.

The Java and C# language allow to define the *main* method inside the class which helps the translation of the algorithm through static methods while the C++ language only allows to define the model needing the *main* function to create an object to be executed.

3) Web programming language

The programming languages for the Web, as the name implies, are intended to develop applications that run on the Web. The web is composed of a set of servers that provide services to clients that are represented by browsers. Algorithmi provides the PHP languages, which is used to develop server-side applications, and the JavaScript language used to develop client-side applications, the browser. Alg. 5 presents the translation of the algorithm for the PHP and Javascript languages.

ALG. 5 - ALGORITHMO TRANSLATIONS REPRESENTED IN ALG. 1 FOR THE PHP AND JAVASCRIPT LANGUAGES.

```
<?php
function main() {
    $txt = "Hello ";
    echo "What's your name ? ";
    $name = isset($_POST["$name"])?
$_POST["$name"] : isset($_GET["$name"]) ?
$_GET["$name"] : 0;
    echo $name . "<br/>";// print input
    echo $txt . $name . "!";
}
main();
?>
```

```
function main() {
    var txt = "Hello ";
    document.write("What's your name ? ");
    name = prompt("name","");
    document.write(name + "\n");
    document.write(txt + name + "!");
}
```

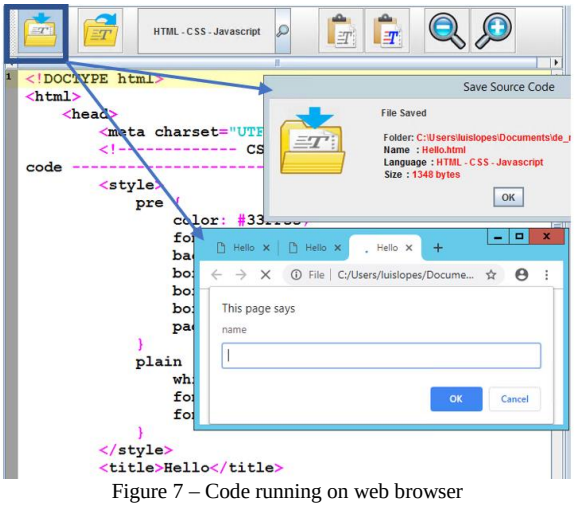

Figure 7 – Code running on web browser

These languages are used to do data processing and whose result is presented to the user using Hyper Text Markup Language (HTML) languages, which will be formats through Cascading Style Sheets (CSS). Combining HTML, CSS and Javascript it is possible to run the algorithm in the student's browser and do a demonstration of the automatic translation for high level languages. With a simple click Algorithmi writes the program and executes it in the web browser as we can observe in Figure 7.

The result of running the program is shown in Figure 8.

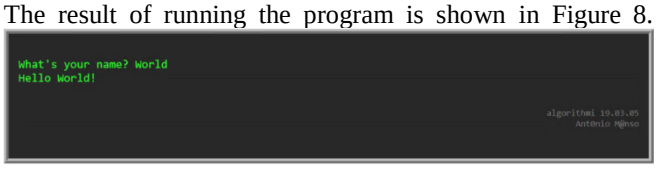

Figure 8 – Writing on the screen by the program using Javascript

IV. Conclusion

In this article we introduce the translation functionality with a learning environment system. Algorithmi introduced English and Portuguese natively in the system, which function as standard languages and we introduced Chinese using Google's Application Programming Interface (API) as a demonstration of concept, to show the use of the system with non-Western characters. Since none of the authors dominates the Chinese language, we cannot guarantee that the translation is correct. As future work, we want to ask the community for help, so that people can translate it into their native language.

Algorithmi is under development with its functionalities and usability being tested during the 2016/2017 and 2017/2018 school years. The feedback of students through programming with the Portuguese language has been very positive in teaching programming logic.

The translation into the programming languages helps the algorithmic language transaction for the computational programming languages that are used in more advanced programming classes.

In the future we plan to introduce new programming models, such as functional languages (lambda) or new languages such as Swift.

References

[1] M. Butler e M. Morgan, "Learning challenges faced by novice programming students studying high level and low feedback concepts", ASCILATE 2007 Singapore, 2007, pp. 99-107.

[2] T. Jenkins, "On the difficulty of learning to program", Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Science, 2002, pp. 27-29.

[3] E. Lahtinen, K. A. Mutka e H. M. Jarvinen, "A Study of the difficulties of novice programmers", Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer ITiCSE'05, 2005, pp. 14-18.

[4] A. Yadin, Reducing the dropout rate in an introductory programming course. ACM Inroads, 2(4), 2011, pp. 71-76.

[5] H. Abelson, N. Goodman, e L. Rudolph, "LOGO Manual", Massachusetts: Artificial Intelligence Lab, Massachusetts Institute of Technology, 1974.

[6] MIT Media Lab, "Scratch", Accessed 17 November 2017 at https://scratch.mit.edu/

[7] A. Manso, L. Oliveira, e C. G. C. Marques, " Ensino da Programação através da Linguagem Algorítmica e Fluxográfica", In C. V. Carvalho, R. Silveira & M. Caeiro (Coord.), TICAI 2009. TIC's para a Aprendizagem da Engenharia. IEEE, Sociedade de Educação: Capítulos Espanhol e Português. Porto: Edições Politema, 2011, pp. 105-110.

[8] A. Manso, C. Marques e P. Dias, "Portugol IDE v3.x: A new environment to teach and learn computer programming", In M. C. Gil, E. T. Caro, M. E. Auer e M. P. B. Merino (Cords.), IEEE EDUCON 2010 Conference - The Future of Global Learning in Engineering Education. Madrid: UPM - Servicio de Publicaciones - EUI – UPM, pp. 1007-1010.

[9] S. R. Fischer, "História da Escrita", London: Reaktion Books, 2003.

[10] B. J. Maclennan, "Principles of Programming Languages. Design, Evaluation and Implementation", Oxford: Oxford University Press, 1999.

[11] Python Software Foundation, PEP 20 - The Zen of Python, Accessed 04 April 2019, https://scratch.mit.edu/https://www.python.org/dev/peps/pep-0020/