Impact of Calango language in an Introductory Computer Programming Course

Geovana Silva
Educational Technology Laboratory
University of Brasília (UnB)
Brasília, Brazil
geovana.ramos@aluno.unb.br

Giovanni Santos
Department of Electrical Engineering
University of Brasília (UnB)
Brasília, Brazil
giovannix@unb.br

Edna Dias Canedo
Department of Computer Science
University of Brasília (UnB)
Brasília, Brazil
ednacanedo@unb.br

Vandor Rissoli
Educational Technology Laboratory
University of Brasília (UnB)
Brasília, Brazil
vandorissoli@gmail.com

Bruno Praciano
Department of Mechanical Engineering
University of Brasília (UnB)
Brasília, Brazil
bruno.justino@ieee.org

Guilherme Andrade
Educational Technology Laboratory
University of Brasília (UnB)
Brasília, Brazil
guilhermeandrade@aluno.unb.br

***Abstract*—This Research Full Paper studies the impact of using Calango language in the introductory learning of algorithms and computer programming by Brazilian engineering undergraduates. Calango is an educational multi-platform tool designed to facilitate algorithm learning and provide a simple syntax that enables developing advanced logic, so that students concentrate on logic without worrying about language details, considering its based on Portuguese pseudocode commands. A survey was conducted to evaluate if the teaching methods and tools used in the engineering introductory computer programming course were helpful. The questionnaire was applied at the end of four consecutive semesters of lecturing the course, and five questions were selected to analyze Calango's influence in the learning process. Approximately 95% of the participants agreed that Calango should continue in the class. Students reported that the learning of C language was more accessible due to the initial contact with Calango, mainly because they could write code in Portuguese, their native language. Overall results show that Calango was well-received by students, and it eased their early programming learning.**

***Index Terms*—Introductory Programming, Pedagogy, Engineering Education, Calango.**

I. Introduction

Learning the basics of algorithms and programming is part of an Engineering curriculum [1]. Besides learning a programming language, students also have to develop problem-solving abilities [2], but introductory courses do not spend much teaching time on logical thinking [3]. Many engineering students do not use the language they are learning in the introductory programming course but only can solve a problem using a computer. Therefore, when introducing programming, classes must make use of languages that do not create a barrier in syntax and semantics, so that students can focus on developing abilities to solve problems using algorithms.

Brazilian engineering students face this problem when learning primary programming concepts, mainly because, beyond familiar syntax and semantics difficulties that any programming student faces [4], non-native speakers of English have questions remembering programming keywords [5]. The top ten programming languages of 2016 work with English-written commands [6], and programming languages chosen by introductory courses are no different [7].

According to Watson and Li [8], Portugal, Germany, and Brazil had the highest mean failures rates among introductory programming courses worldwide that ran in sometime between 1979 and 2013. These countries do not have English as a native language, and two of them have Portuguese as the official language.

This work intends to understand the impact of using a programming language with commands closer to pseudocode and written in the student's native language, including an interface that allows code tracing and debugging easier to students concentrate more on solutions than on how to write them. In this perspective, students can use the knowledge already established in their cognitive structures to solve the proposed activities and expand their knowledge through meaningful learning [9].

The language used is called Calango[1], an educational multi-platform tool designed to ease algorithm learning and prepare learners before introducing C language. The course chosen to test Calango has been taught using C language for many years, and engineering students take it. During four consecutive semesters, Calango was introduced before C at the end of each semester, students responded to questions that approached their opinions on the course teaching methodology. Five survey questions (SQ) were selected to evaluate Calango, and each one assesses a different aspect of the teaching methodology.

Worldwide introductory programming courses have a mean rate failure that is around 30%, and Brazilian courses failure rates are mostly above the worldwide one [8], which is higher than the grades of other bachelor courses [10]. Also, students that pass without a firm grasp of basic concepts present

[1]https://cae.ucb.br/calango/

difficulties in the subsequent classes that rely on these skills [11]. If the results collected are positive, it indicates that students feel comfortable learning with Calango, which can improve their knowledge.

The paper is organized as follows: Section II introduces important concepts related to this work; Section III discusses other contributions as the one presented by this paper; Section IV contextualize the methodology to collect the survey data; Section V presents data results and implications; Section VI presents threats to validity of this research; and Section VII concludes with overall work done and final outcomes.

II. Background

A. *Student's difficulties in Introductory Programming*

Learning to program effectively requires a set of abilities such as problem-solving, critical thinking, creativity, math, and basic English [12]. Problem-solving and critical thinking are the most important because the main goal when developing a software or program is to identify a necessity and instruct a computer to come up with the answers. As introductory courses fail to build a solid problem-solving ability in students, teachers mentioned that students have a hard time dealing with problems from the real world using a programming language [13].

Apart from personal problems and environment context, students may drop an introductory programming course because they become frustrated when they do not understand what they are dealing with and can not design a solution [14]. Some courses introduce algorithms directly in a programming language. Therefore program design and language syntax become intertwined [15], making learning more difficult. This teaching method is misguided because students that can read and trace code cannot necessarily solve problems [16].

Besides the logical thinking and syntax barriers, difficulties also appear when tracing and debugging code, because it requires understanding a programming language well enough to find and fix bugs [17]. Portuguese professors stated that they do not like to use a debugger in class because it adds more complexity to the learning process with reduced results [18], while another research proved that a tool with custom error messages for novices was more helpful than the standard Java error messages [19].

The syntax is an even more significant barrier for non-English speakers novice programming learners. In [20], Portuguese speakers reported many difficulties related to keywords and their translation, as it did not represent well in their native language what is the keyword function, and sometimes the translation does not even exist. One Portuguese speaker said that ”it is necessary to have at least intermediate English. Otherwise, you will have many problems trying to understand the way programming works". Other non-English speakers interviewed also reported that learning English and programming at the same time was hard.

Despite the difficulties presented, advanced programming relies heavily on English, such as documentation and communication with the programming communities. On the other hand, it is unnecessary to add more complexity to an introductory programming course for students of a variety of engineering fields, mainly because not all students will be programmers by profession. Thus, an intermediary programming language is welcome in this case. The necessity of a programming language in novices' native language is evident in the research of Olatunji, Oladosu, Odejobi and Olabiyis [21], in which 89% of the respondents would like to program in their native language, and most of the contrary responses are from respondents that already have programming knowledge.

B. *Pseudocode*

An algorithm is a well-defined sequence of execution instructions on a set of generated input and output values. When solving a problem using computer programming, algorithm development happens before writing code. Algorithm thinking can be designed in many forms, and one of them is pseudocode, which is "an outline of a program, written in the form of spoken language using common words that can easily be converted into real programming statements" [22].

Pseudocode has no standard and, therefore, it is a great tool to promote logical thinking as there is no concern towards syntax and semantics. However, many students bypass using it in the solution development process and move directly to coding because pseudocode is not executable and direct feedback of an algorithm design cannot be obtained [23]. Also, depending on how the student structures his solution, the transition to a computer program as code can be challenging to achieve.

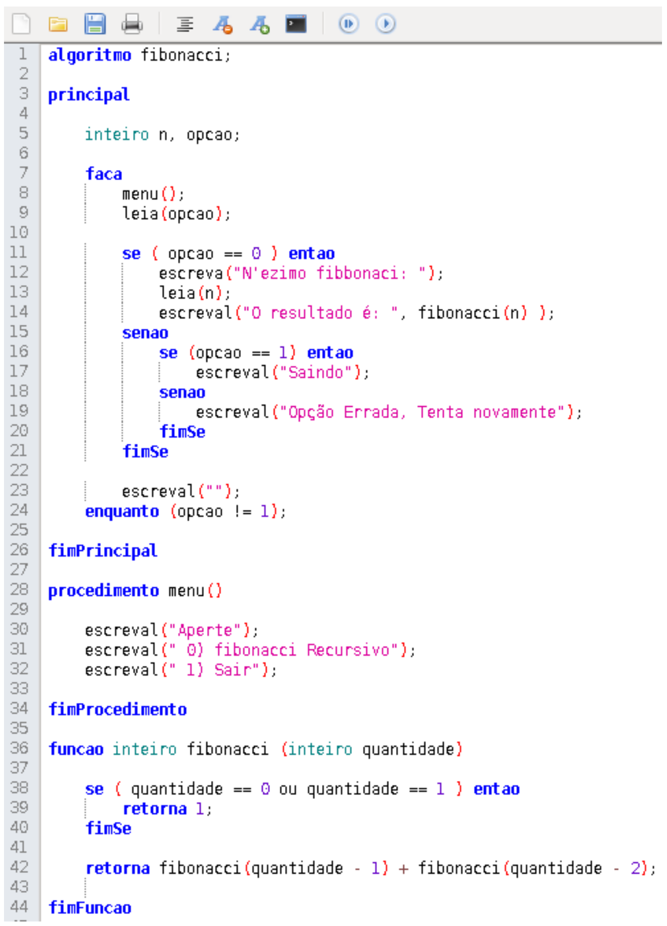
C. *Calango*

Calango is a tool to support the construction of algorithms. It runs Portuguese-written pseudocode as if it were a computer program, improving the understanding of how a program works. Other advantages of teaching introductory programming with Calango are:

- It is easier to learn for Portuguese-speaking learners that do not understand English, as the most popular programming languages receive English-written commands.
- Provides a simple syntax that does not limit algorithm development so that students will concentrate on logic to solve a problem without worrying about language details.
- Allows the execution of the algorithm step by step, allowing the student to follow what is being done in each instruction and assisting him in the code debugging process.
- Displays the content of each variable present in the algorithm, allowing the student to monitor changes in values that occurred during execution.
- As it is a tool focused on the educational aspect of using algorithms, errors present at compile time are presented in a more instructive way.
- It is a multi-platform tool, allowing students to use it in any operating system.

Calango was based on C language because of its notoriety and for being widely used as a first programming language [7].

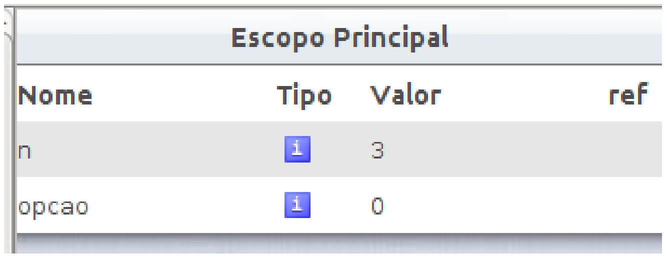
Therefore, Calango works best in introducing programming in courses that will later teach the C language. Fig. 1 shows how a recursive Fibonacci program looks like written in Calango. The example has many important programming structures such as loops, conditions, functions, user input, and console output.



```
 1  algoritmo fibonacci;
 2
 3  principal
 4
 5      inteiro n, opcao;
 6
 7      faca
 8          menu();
 9          leia(opcao);
10
11          se ( opcao == 0 ) entao
12              escreva("N'ezimo fibbonaci: ");
13              leia(n);
14              escreval("O resultado é: ", fibonacci(n) );
15          senao
16              se (opcao == 1) entao
17                  escreval("Saindo");
18              senao
19                  escreval("Opção Errada, Tenta novamente");
20              fimSe
21          fimSe
22
23          escreval("");
24      enquanto (opcao != 1);
25
26  fimPrincipal
27
28  procedimento menu()
29
30      escreval("Aperte");
31      escreval(" 0) fibonacci Recursivo");
32      escreval(" 1) Sair");
33
34  fimProcedimento
35
36  funcao inteiro fibonacci (inteiro quantidade)
37
38      se ( quantidade == 0 ou quantidade == 1 ) entao
39          retorna 1;
40      fimSe
41
42      retorna fibonacci(quantidade - 1) + fibonacci(quantidade - 2);
43
44  fimFuncao
```

Fig. 1. Example of a program that gives the n-th Fibonacci number in Calango.

The interface allows running the whole program or executes step by step. If the later was chosen, the panel is shown in Fig. 2 would appear on the right side of the window, and the student could follow the values of the variables during each step of the execution. For both execution modes, a console appears to the student.

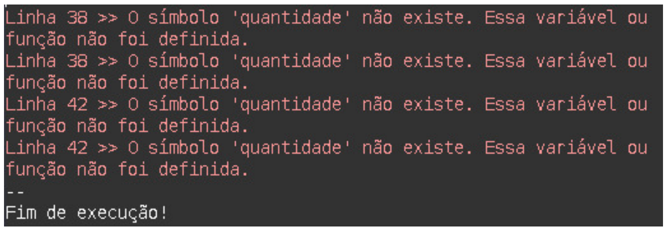


Escopo Principal

Nome	Tipo	Valor	ref
n	i	3	
opcao	i	0	

Fig. 2. Panel from Calango interface for code tracing.

In the Fibonacci example shown in Fig. 1, if the student had forgotten the variable *quantidade* as a parameter in the *fibonacci* function, an error message as shown in Fig. 3 would appear on a console. The message is in Portuguese, and for each error, the console shows the exact line explicitly.



```
Linha 38 >> O símbolo 'quantidade' não existe. Essa variável ou
função não foi definida.
Linha 38 >> O símbolo 'quantidade' não existe. Essa variável ou
função não foi definida.
Linha 42 >> O símbolo 'quantidade' não existe. Essa variável ou
função não foi definida.
Linha 42 >> O símbolo 'quantidade' não existe. Essa variável ou
função não foi definida.
--
Fim de execução!
```

Fig. 3. Error message from Calango.

Besides comments and operators' keywords, Calango has a translated keyword for the most critical and basic commands in C. Also, it does not require braces, and the end of a procedure is marked by another keyword, as shown in some examples from Table I.

TABLE I
CORRESPONDING CALANGO KEYWORDS FOR C KEYWORDS

C keyword	Calango keyword
Main	

int main(void){...}
int func()
return
	principal ... fimPrincipal
	funcao func()
	retorna

Logical operators	
&& || !	e ou nao
Data types	
char* char double int 0, 1	texto caracter real inteiro logico
Read and write	

printf("...")
scanf ("...", var)
	escreval("...")
	leia(var)

Condition	

if()...
else
switch()...
case
break
	se() entao ... fimSe
	senao
	escolha ... fimEscolha
	caso
	interrompa

Loops	

while()
do ... while
for(int i=0; i<=3; i++)
	enquanto() faca
	faca ... enquanto
	para (i de 0 ate 3 passo 1)

III. Related Work

Introductory programming courses are known for having high failure rates. Works considering courses all around the world found failure rates around 30% [24], [25], and some works found that Brazilian failure rates were around 50% [8], [10]. To decrease these rates, it is necessary to understand the factors that impact students' learning process and develop new methodologies to deal with their difficulties in introductory programming.

One of the fields of investigation related to introductory programming courses is motivating students in the learning process to drop the high failure rates. Maiga and Emanuel [26] approached the issue using Gamification to encourage learners to learn Java. The study received positive feedback from students as they became enthusiastic about learning with Gamification. Bandeira et al. [27] evaluated students learning and motivation when using the Competitive Programming methodology.

Besides methodologies to improve engagement, tools were also developed to achieve the same result. An Educational Board was developed using Arduino to enhance the attractiveness of the course lectured to first-year undergraduates of Computer Science [28]. Although students liked the methodology, it does not seem to attack another programming problem beyond engagement. Another approach is the use of visualization tools, like Willow [29], which provides visualization of data structures and turns to learn more interactive and eases the comprehension of abstract concepts.

Barrera [30] studied the competencies that should be developed in an introductory programming course and the methodologies and strategies that must be used in the teaching process. This work highlights the importance of engaging the students in the learning of basic programming concepts and the necessity and impact of developing abilities beyond language syntax using modern methodologies. It was also clear the importance of developing problem-solving skills and logical thinking, those being the legacy of introductory programming to further programming courses.

Considering the difficulties faced by novices and presented at Section II-A, many works tried to develop an environment favorable to the development of logical thinking. Olsen [31] changed the sequence of course topics introducing logical thinking first with pseudocode instead of beginning the course with C++ syntax. Although results were positive, students did not have a tool that could run their pseudocode, which is essential when learning how computer programs work and smoothing the transition to C++ language.

From another standpoint, but with the same hypothesis, another work used two custom-developed programming languages close to natural language in class and compared them [32]. Each group of students used one of the languages to interact with the game developed for the experiment. One programming language was close to the natural Portuguese language, and the other followed a classical context-free grammar notation, with looks of pseudocode. The experiment was supported by the use of a platform to edit and debug code that was also developed by the authors. The languages are not broad because they only cover commands from the game.

Another approach that is more likely to smooth transition to C language is Block-C [33], which works with both the block and text representation of C language. Although the method reduces syntax errors when students program in C, it does not assess the problem-solving abilities. Also, students consider the block-based approach more lengthy to code, more distant from real-world programming languages, less potent than the text-based approach [34], and does not make it possible to experience program debugging.

With that in mind, other works proposed tools such as PROBSOL [35] and LPL [36] that focus on problem-solving with the text-based approach. PROBSOL does not require coding, only reading pseudocode and ordering the steps to solve a problem. Therefore, it is not executable by the student. LPL receives students' code, translates to C and C++, and has a simple debug box, but no tracing code feature. Meanwhile, another work focused on developing a tool called Gauntlet [37], focusing only on turning debugging for novices easier. All these tools require basic English knowledge.

These tools proved to be useful in each work, but they probably would not be as effective in classes with students that are not native English-speakers. Therefore, some researches were done with pseudo languages that were developed in the students' native languages as an Arabic version of LISP and SQL [38] and also a Spanish pseudo-language for novices to build interfaces in Java [39].

IV. Methodology

A. Research Context

The students who participated in this research are undergraduates at Faculty UnB Gama (FGA), one of the campuses of the University of Brasília (UnB). This faculty is ideal for the experiment because it only offers Engineering bachelor's degrees, which are: Software, Aerospace, Energy, Electronics, and Automotive. All students take the Algorithms and Computer Programming (ACP) course in early semesters as all degrees further require some level of knowledge in programming, especially Software Engineering [40].

The course is mandatory to all degrees and is mostly taken by first semester undergraduates as suggested by the University, although it is possible to take it at a later time. Software Engineering students do not tend to postpone it, not even one semester, as it is a prerequisite for most subsequent courses. Other students postpone it because, for them, the course is seemed as tricky, therefore it is essential to have a teaching methodology for all student profiles.

A questionnaire was applied at the end of four consecutive semesters, obtaining a simple random sampling involving 264 participants. This research instrument was prepared by a multidisciplinary team that includes teachers in Computer Science, Pedagogy, Mathematics, and Psychology. It was inspired by SEEQ (Students' Evaluation of Educational Quality) and seeks to ascertain the level of student satisfaction and the efficiency of the educational process carried out [41]. It consists of 19 questions covering five subject matters to be observed from the perspective of each student participating in the research: I) personal characteristics of the participant, II) technological resources used for educational support, III) pedagogical posture, IV) student tutoring, V) own analysis with self-assessment.

All students enrolled in the ACP course were invited to participate, but not all of those that started the class responded to the survey as it is possible to drop the enrollment until a certain period of the semester defined by the University. Therefore, the number of students corresponds to the number of students who finished the course successfully. Also, not all questions were answered by all participants, as some of them were not mandatory or were dependent on a previous one.

Even though it is by default, an in-person course, tools that support distance learning were used to keep track of students' learning and to automate code corrections, one of these tools is an Intelligent Teaching Assistant called *Sistema de Apoio Educacional* – Educational Support System (SAE) that "seeks an evaluation more accurate and coherent with the course content given the expectation of specialists that accompany it during teaching-learning" [42].

In light of this, SAE analyses students' performance in each course content and classify their learning as "Satisfactory" or "Unsatisfactory" through Fuzzy Logic based on the frequency of activity completion and their results [43]. At the end of the course, if the student received a "Satisfactory" classification in all course contents, he gets a Significant Score that will be an essential asset when evaluating student's learning in this research. The Significant Score is a result that takes into consideration the student's performance in every course content, and it is a means of fuzzy results from all materials [44].

The Significant Score was chosen rather than grades because it is calculated by a platform that considers a broad number of factors related to the students' learning and performance in each course content that grades would not contemplate. These factors include quantity and result of questions answered besides tutoring usage. For the course chosen in this research, the Significant Score goes from 0 to 10, and a considered great result must be above 5.

Lastly, so that results would not be affected by particularities of different professors and their pedagogical methods, the same professor was responsible for all the classes evaluated, and the course followed the same syllabus in each semester. The teacher's educational posture is based on the proposal of the Theory of Meaningful Learning, which recognizes the importance of the association between the knowledge established in the learner's cognitive structure and the new content presented to him [45].

B. Survey Questions

The questionnaire was applied through an open-source survey software called LimeSurvey, and results were extracted to PDF files for later analysis. Among the questions available in this instrument, five were selected, as shown in Table II. They address the teaching methodology, the technological resources used for educational support during the learning process, and the students' self-assessment. Selected questions were renumbered, and every item can show a different aspect of the impact of introducing programming with Calango. Therefore column "Metric" denotes what can be inferred with the results of each question.

The performance metrics are essential to evaluate three aspects of the experiment: effort to learn, learning efficiency, and use of the methodology. The attempt to learn enables the participants to give coherent answers about the teaching methodology, as they strive to complete all planned activities. If a student did not engage in all activities, he did not participate in the methodology applied as a whole.

Considering only participants that did all the activities proposed, it is necessary to analyze their results to evaluate if they were successful or not in the learning process. Instead of using only raw grades, a more wide-range evaluation will be done, and the Significant Score from SAE will be used to classify the learning of those who completed all activities, as it considers tutoring usage, activity completion, and results. The Significant Score is collected through SQ.1.

Also, it is necessary to question students if the professor was committed to easing learning in their perception using a methodology. For this aspect, SQ.2 will provide data to conclude if the professor could use the methodology from the students' point of view. At last, with all these research variables set, students themselves will provide feedback about Calango and the course, subjectively in SQ.4 and objectively in SQ.3 and SQ.5.

V. Results

A. Survey Question 1: Indicate what was your Significant Score assigned by SAE.

This question helps us measure the learning of students that followed and accomplished all activities and therefore got a Satisfactory classification in all course contents because the Significant Score is only calculated under these conditions. Table III lists the number of students that were able to answer the question and Fig. 4 (a) show percentage numbers. Students who did not meet these conditions would interfere in results because a teaching methodology efficiency depends on students' effort to learn.

Students that concluded all course contents met SAE conditions. These students had to see their Significant Score in the platform and write it into the form. Table IV shows the result for each semester and a final balance. Of the 90 students who completed all course activities and met conditions, 88 provided an answer to SQ.1.

Considering that the course grade range is 0 to 10, the passing grade is five, and that the Significant Score shown to students is considering this, the final mean is a great result, which implies that students that completed all activities proposed by the course had a satisfactory result in them. The maximum result shows that some students were able to achieve the best result possible in the course.

B. Survey Question 2: Would you say that your teacher is using some methodology of learning or no method is being used by him in conducting this course?

Concerning students' perception of the methodology applied, SQ.2 can indicate how well the professor handled the course. A mostly positive number of answers to this question indicates that students perceived an effort to ease their learning, and therefore the course under analysis was taught as

TABLE II
SELECTED QUESTIONS FROM SURVEY.

Identifier	Question	Answer Type	Metric
SQ.1	Indicate what was your Significant Score assigned by SAE.	Number	Students' performance
SQ.2	Would you say that your teacher is using some methodology of learning or no method is being used by him in conducting this course?	Multiple-choice	Professor's performance
SQ.3	Would you recommend your other colleagues to take this course as you did this semester, in this form of teaching-learning, using these technological support resources?	Multiple-choice	Calango's approval rate
SQ.4	Were the code development tools adopted by this course important for your learning? What were your impressions about learning Calango before C?	Open-ended	Calango's approval rate
SQ.5	In your opinion, this course met your initial expectations for this semester?	Multiple-choice	Course's approval rate

TABLE III
STUDENTS THAT CONCLUDED ALL COURSE CONTENTS

Year	Semester	Yes	No
2018	1	9	14
2018	2	11	13
2019	1	44	7
2019	2	26	14
Total		90	48

TABLE IV
STUDENTS' SIGNIFICANT SCORES

Year	Semester	Answers	Minimum	Maximum	Mean
2018	1	9	6.00	8.00	6.72
2018	2	11	0.00	10.00	5.98
2019	1	43	4.00	9.00	6.77
2019	2	25	0.00	10.00	6.76
Total		88	0.00	10.00	6.56

it should. Table V shows the number of students that perceived or not that a teaching methodology was being used and Fig. 4 (b) shows final percentage results.

TABLE V
STUDENTS THAT PERCEIVED A TEACHING METHODOLOGY

Year	Semester	Answers	Yes	No	Did not know
2018	1	31	28	1	2
2018	2	28	28	0	0
2019	1	66	62	1	3
2019	2	57	53	0	4
Total		182	171	2	9

From 182 answers, 94% stated clearly that a methodology was applied. Therefore, it is safe to conclude that the professor followed the course methodology to further evaluate students' perception of Calango.

C. Survey Question 3: Would you recommend your other colleagues to take this course as you did this semester, in this form of teaching-learning, using these technological support resources?

At last and most importantly, it is necessary to question students if they approved the use of Calango, and this was measured in two different ways. To measure directly and objectively, SQ.3 had multiple-choice answers, and its results are listed in Table VI and Fig. 4 (c).

TABLE VI
STUDENTS THAT WOULD RECOMMEND THE METHODOLOGY USED

Year	Semester	Answers	Yes	No
2018	1	31	29	2
2018	2	28	28	0
2019	1	66	63	3
2019	2	57	53	4
Total		182	173	9

The previous questions had the goal to show that the teaching environment was ideal to evaluate Calango, but SQ.3 is the most important one as it assesses Calango directly and objectively. Approximately 95% of the participants agreed that Calango should continue in the course, that is, they were so satisfied with the experience that students from further semesters should also experience it.

D. Survey Question 4: Were the code development tools adopted by this course important for your learning? What were your impressions about learning Calango before C?

Considering SQ.4 was open-ended, it goes beyond understanding if students liked Calango because it also makes it possible to identify why they liked it or not. Every answer was read and classified. First, answers were read without classification to analyze what were the overall impressions of the students.

Once the most common statements were identified, another round of reading was made to count how many students quoted these common impressions. To avoid misinterpretations, only answers that explicitly expressed a statement or opinion were considered when counting. Also, as the question asked about the other tools used in the course as well, only the ones that explicitly cited Calango were counted.

This question received 184 answers from which 147 explicitly addressed Calango. Approximately 83% of answers about Calango were overall positive. In most of these positive feedbacks, many students informed that Calango reinforced logical thinking and that transitioning to C language was not difficult due to Calango. Two different students provided the following statements:

> *"Calango was one of the most important tools, as it introduced the necessary logic and made the C language much easier to use because it is very similar."*

"Learning logic first through Calango was extremely useful. Once you learn logic, it is much easier to program in other languages. When comparing my class performance with others, I realized the difficulty that my colleagues from other classes faced when writing code that my class had already done more than a month ago."

Many positive opinions also included statements that associated the importance of Calango in teaching learners with no previous contact with programming, and many identified themselves as novices. Also, around 8,2% of the positive answers reported that programming in their native language was important when learning initial concepts, in this case, Portuguese. Two different students provided the following statements:

"I believe that learning the logic of programming, starting in Portuguese (with Calango), and then going to C language is essential for those who had never been in contact with any programming language (like me)."

"Calango undoubtedly helps a lot by introducing programming logic, for those who have never told them it is of great importance to become familiar with the terms first in Portuguese and then work on them in English."

Some students that provided mostly negative feedback had already been in contact with programming, and therefore Calango was not important for them to acquire more knowledge, although it did not hinder learning, as shown by the following statement:

"This semester, for those who already had programming experience, learning Calango was a small but unnecessary burden; for those who didn't, it contributed a lot before moving on to C language"

A few negative feedbacks included comments about some difficulty in certain commands that were not so similar to C as others, after the transition. It may be a problem for novices, but this is a common difficulty for programmers transitioning to new languages. Other suggestions to Calango were the possibility to see the corresponding commands in C and an online judge as the one used for C in the course.

E. Survey Question 5: In your opinion, this course met your initial expectations for this semester?

In SQ.5, students were questioned about their expectations regarding the course. Besides aligning course content with students learning desires, results from this question are also important to quantify students' satisfaction with a methodology that includes Calango. Table VII shows the results in numbers and Fig. 4 (d, e) shows percentage results.

The course met expectations for 76,4% of the students that participated in this question and that were also satisfied. If we separate satisfaction from expectation, 85.2% of students were satisfied, and 85.7% had their expectations met.

TABLE VII
SQ.5 RESULTS

Year	2018		2019		Total
Semester	1	2	1	2	-
Yes, and I am SATISFIED	16	22	54	47	139
Yes, but I am UNSATISFIED	5	1	5	6	17
No, and I am UNSATISFIED	5	1	3	1	10
No, but I am SATISFIED	5	4	4	3	16
Total					182

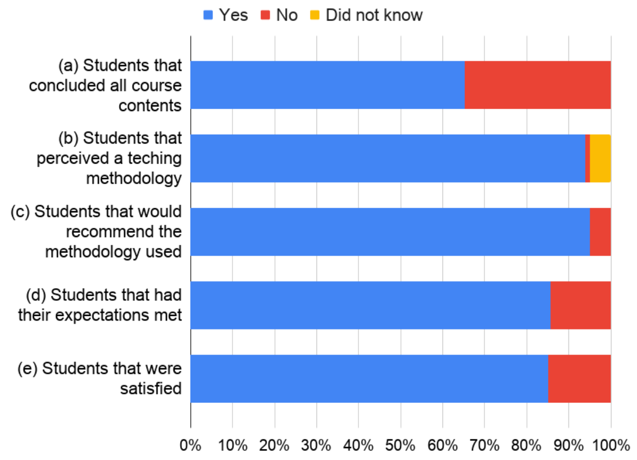


Fig. 4. Percentage results.

VI. THREATS TO VALIDITY

Not all students that took the course answered the survey. This situation includes students that opened the survey and did not answer specific questions; students that finished the course successfully or not but did not open the survey having no overall participation; and students with leave of absence. Although SQ.1 imposed a condition that limited answers, the other survey questions did not have any condition, and even so some students did not answer them.

Regarding commitment in answering correctly, in SQ.1 a value that is provided by a system was requested to the students and the values received by us may not be exactly equal to the ones provided by the system. Also, answers may be affected by student-professor relations and remembrance of classes. Even though results include four different semesters, profiles may impact this research, as the university reserve a certain number of vacancies for freshmen and classes vary from almost 90% of vacancies reserved for freshmen to none, and this information was not collected.

Also, SQ.4 analysis may suffer from misinterpretation even if it relies only on explicit opinions, whether because of the researcher's interpretation or if the student did not express his opinion entirely. The last is the most impacting because a poorly formulated answer still can express an explicit incorrect answer. Lastly, students' opinions about a methodology may not assess well their actual learning [46].

VII. Conclusion

This work is essential to understand if novice learners well accept a tool that supports programming using pseudocode and provides keywords and error messages in students' native language. Four aspects were evaluated by the four survey questions presented, with two of them focusing on the commitment to the methodology of the individuals involved and two gathering direct feedback from students. Results show that from 182 respondents, 95.1% stated objectively that they would recommend the method used in the course, and 85% were satisfied with the class.

The positive opinions collected from the open-ended survey question addressed problem-solving reinforcement, the ease in transitioning to C, and the ease of learning programming with keywords in the native language. Most of the negative opinions stated that it would be better to start with C, but Calango did not compromise the learning. These kinds of answers were expected as it is probable that some students in the class were retaking the course and had the previous contact with programming.

The high number of students reporting ease to transition to C is expected to be a result of having them programming in their native language, that is, their knowledge acquisition was supported on previous knowledge. Other novice tools also try to offer facilitated learning, but they end up distancing from real-world programming, while Calango provides native language keywords that are based on a traditional programming language. Therefore, students rely on what they already know to learn something close to their final goal.

Overall results show that students had a good experience in the learning process with Calango. Also, it is possible to infer that novice learners have an interest in tools that ease learning but simulate real-world programming experience. Although this work is based mostly on perceptions and not learning results, having students interested and motivated when using the programming tools is the first step to better prepare them in introductory programming courses.

In both positive and negative opinions, students also suggested improvements in Calango that can be addressed in future works. Additionally, students still have difficulty in testing and validating the correctness of the implemented code. Many of these programs fail in boundary testing, for example. Thus, future works also include the development of a web-based online judge capable of testing the correctness of applications developed using Calango and assessing the impact of this online judge on improving the quality of programs produced by students.

Acknowledgment

The authors would like to thank the support of the cooperation projects with the Ministry of the Economy (grants DIPLA 005/2016 and ENAP 083/2016), and the Administrative Council for Economic Defense (grant CADE 08700.000047/2019-14).

References

[1] D. Davenport, "Experience using a project-based approach in an introductory programming course," *IEEE Transactions on Education*, vol. 43, no. 4, pp. 443–448, 2000.

[2] Y. Li1, J. Niu1, J. Zhang1, and R. Hao1, "Study of engineering-oriented teaching method in c programming course based on emerging engineering education," in *FIE 2019: 41th Frontiers in Education Conference*, 2019.

[3] S. I. Malik and J. Coldwell-Neilson, "A model for teaching an introductory programming course using adri," *Education and Information Technologies*, vol. 22, no. 3, pp. 1089–1120, 2017.

[4] S. Malik, "Improvements in introductory programming course: Action research insights and outcomes," *Systemic Practice and Action Research*, vol. 31, no. 6, pp. 637–656, 2018.

[5] A. K. Veerasamy and A. Shillabeer, "Teaching english based programming courses to english language learners/non-native speakers of english," *International Proceedings of Economics Development and Research*, vol. 70, p. 17, 2014.

[6] L. Gu, N. Yan, and Y. Xiu, "Discussion on teaching methods and choice of programming language on software engineering major," *DEStech Transactions on Engineering and Technology Research*, no. apetc, 2017.

[7] O. Ezenwoye, "What language?-the choice of an introductory programming language," in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–8.

[8] C. Watson and F. W. Li, "Failure rates in introductory programming revisited," in *Proceedings of the 2014 conference on Innovation & technology in computer science education*, 2014, pp. 39–44.

[9] D. Ausubel, "The acquisition and retention of knowledge: a cognitive view. 2000."

[10] Y. Bosse and M. A. Gerosa, "Reprovações e trancamentos nas disciplinas de introdução à programação da universidade de são paulo: um estudo preliminar," in *XXIII WEI–Workshop sobre Educação em Informática. Recife, julho.*, 2015.

[11] M. Ford and S. Venema, "Assessing the success of an introductory programming course," *Journal of Information Technology Education: Research*, vol. 9, no. 1, pp. 133–145, 2010.

[12] A. Gomes and A. Mendes, "A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE, 2014, pp. 1–8.

[13] S. M. M. Rubiano, O. López-Cruz, and E. G. Soto, "Teaching computer programming: Practices, difficulties and opportunities," in *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2015, pp. 1–9.

[14] A. Vihavainen, M. Paksula, and M. Luukkainen, "Extreme apprenticeship method in teaching programming for beginners," in *Proceedings of the 42nd ACM technical symposium on Computer science education*, ser. SIGCSE '11. New York, NY: ACM, 2011, pp. 93–98.

[15] S. Garner, "Learning resources and tools to aid novices learn programming," in *Informing science & information technology education joint conference (INSITE)*, 2003, pp. 213–222.

[16] C. Cabo, "Effectiveness of flowcharting as a scaffolding tool to learn python," in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–7.

[17] S. Fitzgerald, R. McCauley, B. Hanks, L. Murphy, B. Simon, and C. Zander, "Debugging from the student perspective," *IEEE Transactions on Education*, vol. 53, no. 3, pp. 390–396, 2009.

[18] A. Gomes, W. Ke, S. K. Lm, A. Siu, A. J. Mendes, and M. J. Marcelino, "A teacher's view about introductory programming teaching and learning—portuguese and macanese perspectives," in *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2017, pp. 1–8.

[19] B. A. Becker, "An effective approach to enhancing compiler error messages," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016, pp. 126–131.

[20] P. J. Guo, "Non-native english speakers learning computer programming: Barriers, desires, and design opportunities," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–14.

[21] E. K. Olatunji, J. B. Oladosu, O. A. Odejobi, and S. O. Olabiyisi, "A needs assessment for indigenous african language-based programming languages," *Annals of Science and Technology*, vol. 4, no. 2, pp. 1–5, 2019.

[22] A. Karatrantou, C. Panagiotakopoulos, and A. Patras, “Algorithm, pseudo-code and lego mindstorms programming,” in *Proceedings of International Conference on Simulation and Programming for Autonomous Robots/Teaching with Robotics: Didactic Approaches and Experiences, Venice, Italy*, 2008, pp. 70–79.

[23] S. Garner, “A program design tool to help novices learn programming,” *ICT: Providing choices for learners and learning*, pp. 321–324, 2007.

[24] J. Bennedsen and M. E. Caspersen, “Failure rates in introductory programming: 12 years later,” *ACM Inroads*, vol. 10, no. 2, pp. 30–36, 2019.

[25] A. Luxton-Reilly, V. V. Ajanovski, E. Fouh, C. Gonsalvez, J. Leinonen, J. Parkinson, M. Poole, and N. Thota, “Pass rates in introductory programming and in other stem disciplines,” in *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, 2019, pp. 53–71.

[26] J. Maiga and A. W. R. Emanuel, “Gamification for teaching and learning java programming for beginner students—a review,” *Journal of Computers*, vol. 14, no. 9, September 2019.

[27] I. N. Bandeira, T. V. Machado, V. F. Dullens, and E. D. Canedo, “Competitive programming: A teaching methodology analysis applied to first-year programming classes,” in *2019 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2019, pp. 1–8.

[28] I. Perenc, T. Jaworski, and P. Duch, “Teaching programming using dedicated arduino educational board,” *Computer Applications in Engineering Education*, vol. 27, no. 4, pp. 943–954, 2019.

[29] P. Moraes and L. Teixeira, “Willow: A tool for interactive programming visualization to help in the data structures and algorithms teaching-learning process,” in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, 2019, pp. 553–558.

[30] Y. C. Barrera, “Skills and strategies to promote learning algorithms and programming. evolution from learning objectives,” in *2017 XLIII Latin American Computer Conference (CLEI)*. IEEE, 2017, pp. 1–10.

[31] A. L. Olsen, “Using pseudocode to teach problem solving,” *Journal of Computing Sciences in Colleges*, vol. 21, no. 2, pp. 231–236, 2005.

[32] O. L. Oliveira, A. M. Monteiro, and N. T. Roman, “Can natural language be utilized in the learning of programming fundamentals?” in *2013 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2013, pp. 1851–1856.

[33] C. Kyfonidis, N. Moumoutzis, and S. Christodoulakis, “Block-c: a block-based programming teaching tool to facilitate introductory c programming courses,” in *2017 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2017, pp. 570–579.

[34] D. Weintrop and U. Wilensky, “To block or not to block, that is the question: students’ perceptions of blocks-based programming,” in *Proceedings of the 14th international conference on interaction design and children*, 2015, pp. 199–208.

[35] S. I. Malik, R. Mathew, and M. M. Hammood, “Probsol: A web-based application to develop problem-solving skills in introductory programming,” in *Smart Technologies and Innovation for a Sustainable Future*. Springer, 2019, pp. 295–302.

[36] M. Naveed, M. Sarim, and K. Ahsan, “Learners programming language a helping system for introductory programming courses,” *Mehran University Research Journal of Engineering and Technology*, vol. 35, no. 3, pp. 347–358, 2016.

[37] T. Flowers, C. A. Carver, and J. Jackson, “Empowering students and building confidence in novice programmers through gauntlet,” in *34th Annual Frontiers in Education, 2004. FIE 2004.* IEEE, 2004, pp. T3H–10.

[38] H. Elazhary, “Facile programming.” *Int. Arab J. Inf. Technol.*, vol. 9, no. 3, pp. 256–261, 2012.

[39] E. E. Aispuro, G. Licea, J. Suárez, A. Sandoval, M. A. Carreño, I. Estrada, R. Juárez-Ramírez, L. Aguilar, and L. G. Martínez, “Supporting the development of interactive applications in introductory programming courses,” *Computer Applications in Engineering Education*, vol. 20, no. 2, pp. 214–220, 2012.

[40] E. D. Canedo, G. A. Santos, and S. A. A. de Freitas, “Analysis of the teaching-learning methodology adopted in the introduction to computer science classes,” in *FIE*. IEEE Computer Society, 2017, pp. 1–8.

[41] D. Watkins, H. W. Marsh, and D. Young, “Evaluating tertiary teaching: A new zealand perspective,” *Teaching and Teacher Education*, vol. 3, no. 1, pp. 41–53, 1987.

[42] V. R. V. Rissoli, L. M. M. Giraffa, and J. de Paula Martins, “Sistema tutor inteligente baseado na teoria da aprendizagem significativa com acompanhamento fuzzy,” *Informática na educação: teoria & prática*, vol. 9, no. 2, 2006.

[43] V. R. V. Rissoli, “Uma proposta metodológica de acompanhamento personalizado para aprendizagem significativa apoiada por um assistente virtual de ensino inteligente,” Ph.D. dissertation, University of Rio Grande do Sul, 2007.

[44] V. R. V. Rissoli, L. M. M. Giraffa, and J. de Paula Martins, “Sistema tutor inteligente baseado na teoria da aprendizagem significativa com acompanhamento fuzzy,” *Informática na educação: teoria & prática*, vol. 9, no. 2, 2006.

[45] D. P. Ausubel, J. D. Novak, H. Hanesian *et al.*, “Educational psychology: A cognitive view,” 1968.

[46] L. Deslauriers, L. S. McCarty, K. Miller, K. Callaghan, and G. Kestin, “Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 39, pp. 19 251–19 257, 2019.