Connections and Influences Among Topics of Learning How to Program

Yorah Bosse
University of Sao Paulo (USP)
Sao Paulo, Brazil
yorah@ime.usp.br

David F Redmiles
University of California (UCI)
Irvine, USA
redmiles@ics.uci.edu

Marco Gerosa
Northern Arizona University (NAU)
Flagstaf, USA
marco.gerosa@nau.edu

***Abstract*—This Full Paper of Innovative Practice research shows results that could help to avoid some challenges faced by those who seek to learn how to program. To help improve learning, educators need a deep understanding of the obstacles students must overcome; otherwise, teaching strategies will be uncertain. Moreover, a shallow understanding of topics learned in introductory programming courses can negatively influence the learning of future topics. With the above motivation, we conducted 16 semi-structured interviews with instructors who teach introductory programming courses and we also collected diaries kept by 110 students during their studies. The qualitative analysis of these data revealed connections between the studied contents such as dependencies. Our analysis shows that many difficulties arise from the incorrect application of the knowledge necessary in learning new content, usually because the student has not learned earlier topics or learned them superficially. The main contribution of this paper is a theory that describes the connections among topics of learning how to program, showing the influence that knowledge about one can have on others.**

***Index Terms*—learning to program, novice learners, barriers to learning, introductory programming, computational thinking**

I. Introduction

Learners of programming around the world in introductory courses inevitably run into frustrating hurdles [1]. One such hurdle that causes frustration in courses is understanding topics [2]–[8]. Although people learn to program in different ways, such as with graphical programming blocks, text, or both together [8]–[10], and for different reasons [11], few of them consider it an easy task [3], [5]. This can be perceived in failure and dropout rates, which reach about 28% among undergraduate students [12]–[14].

"Programming is not an easy subject to be studied" [15] and preparing new generations of professional and casual developers is a big challenge. Even so, it is increasingly necessary to extend programming literacy beyond computer professionals [16]. Knowing how to program is useful in people's autonomy while developing new or adapting existing applications used in their everyday life. In this context, it is important to know the causes that lead students to perform poorly when learning about programming [17] and understand the difficulties faced by them as to avoid frustration, failures, and dropouts, thus helping them to achieve their goal.

Even with a great deal of diverse research in this area, many problems still hold forth and thus we still need to gain a deeper understanding of the challenges faced by students and instructors in teaching/learning programming. In this paper, we contribute with a theory that describes the connections among topics of learning how to program, showing the inuence that knowledge about one can have on others. Understanding those connections is important because students may have difficulty learning new topics if they do not understand well earlier topics. A better knowledge about these connections can give course coordinators and instructors the opportunity to change the way of teaching.

Our theory is grounded on two data sources: the first comprises semi-structured interviews with 16 instructors that have experience teaching introductory programming course (CS1) and the second comprises diaries maintained by 110 students when taking CS1. To guide us throughout the study, we focused on the following research question: What is the influence that the misunderstanding of one topic has on the others?

II. Related Work

We focused our literature review on: pedagogical issues, difficulties learning topics and syntax errors, and general difficulties faced by students and abilities necessary to learn to program, as you can see below.

Pedagogical issues: Computer Programming Education is massive [18] and a lot of studies have being developed to improve programming learning. Pedagogical understanding is important for this improvement, and it is important to give the focus not only on teachers [18], but on students and their process of learning [19]. For that, many studies use constructivism, a well-known learning theory [20]. When working with this learning theory, the focus is more on learning and experiences of the students [20]. Still according to Vann Gorp and Grissom [20], "constructivist classrooms are often viewed as problem-solving environments manifested through three C's: context, construction, and collaboration. Some examples of the use of the constructivism theory are the Contributing Student Pedagogy, where students are encouraged "to contribute to the learning of others and to value

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001 and the first author is supported by UFMS - University of Mato Grosso do Sul to develop her thesis. This work was also funded by FAPESP, through grant 2015/24331-1.

978-1-7281-1746-1/19/$31.00 ©2019 IEEE

the contributions of others" [21] and the study of Effective Pedagogy for CS1 Laboratories [22].

Difficulties learning topics and syntax errors: Some topics covered in CS1 are considered difficult to understand by many novice programmers, such as pointers and abstract data types [23], [24]. Results of studies showed that concepts like repetition, recursion, lists, pointers, passing parameters, abstract data types, and the use of libraries are the most difficult programming concepts [23], [25]. In addition, Sevella and Lee [25] show that students confuse 'For' and 'While' concepts in C and find it difficult to use functions; variables; selection structures, such as writing conditional statements and nested selections. They found no difficulty in understanding lists. However, for the instructors responding to the questionnaire by Piteira and Costa [24], lists are among the most difficult topic, aside from pointers, structured data types, error handling, and parameters. Regarding syntax, Tan and colleagues [17] show that learning programming language syntax is one of the problems faced by students, and Hristova and colleagues [6] report difficulties related to the wrong use of '=' vs '==', '&&' vs '&', '||' vs '|', different amounts of opening and closing parentheses, brackets and quotation marks, wrong separators in 'For' loops, and so on.

General difficulties faced by students and abilities necessary to learn to program: According to some instructors, a constant and intense amount of study is necessary [7]. Another obstacle is that many students overestimate their understanding and do not always see their difficulties [2], [5], [16]. For example, they do not perceive their difficulties understanding issues related to the execution of a program [2] and what happens in the computer memory during this execution [4]. Often the problem is not in learning the concept and its syntax, but rather in appropriately applying it in a program [2], [3]. Students can learn very well, for example, about pointers, but still fail to use them appropriately [2]. The lack of experience of students can also contribute to the difficulties that appear [3], [23]. Other skills cited as necessary for learn to program are computational thinking [26] and the ability to find errors in code [27]. Finding errors in one's own code is also one of the difficulties while learning programming cited by Tan and colleagues [17], as well as designing a program to solve some task. Finally, problems arise from behavior in class, such as lack of participation or hesitation to ask questions, and lack of English proficiency [23].

The results reported in the literature mention difficult topics. Better understanding of these topics is very important and we aimed to understand how these difficulties arise. Interviewing instructors and asking students to keep study diaries, we saw difficulties with topics arising during the study of other topics. We modeled the relationships between the topics, where the connections arose. In the next section, we discuss these findings, comparing with some results presented in the literature.

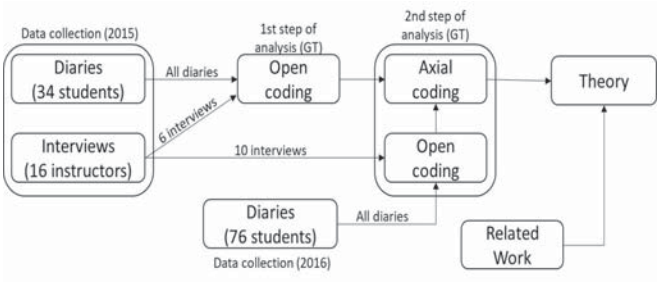

Fig. 1. Research method.

TABLE I
INSTRUCTORS' DATA.

Professor ID	Years of Experience	Gender
P1 - P3	More than 40 years	1 F and 2 M
P4 - P5	31 to 40 years	2 M
P6 - P7	21 to 30 years	2 M
P8 - P12	11 to 20 years	2 F and 3 M
P13 - P16	1 to 10 years	4 M

III. RESEARCH DESIGN

We conducted semi-structured interviews with 16 instructors and asked students to maintain diaries during their studies for the course. Fig. 1 shows an overview of our methodology, which is detailed below.

A. Data Collection

This study is motivated by a previous study conducted at the University of Sao Paulo (Brazil) from 2010 to 2014 based on data about approximately 18,500 students from various majors who enrolled in 29 CS1 [14]. The results showed that approximately 30% of these registrations resulted in failures and dropouts. This percentage corroborates results obtained by Bennedsen and Caspersen [12], [13].

In the present study, we chose to further investigate these courses and topic-related issues. We conducted interviews with instructors to delve into particulars, focusing on the topics to gain a perspective of the students' learning experiences and instructors' teaching experiences [28]. Sixteen instructors from the Computer Science Department at the University of Sao Paulo (USP) were selected for interviews. The Department of Computer Science at USP had more than 260 undergraduates, 304 postgraduates and about 40 teachers who worked with classes in this and other departments. About 30 instructors from the department had taught this course at least once. The first 6 instructors interviewed were those who were teaching introductory programming for six classes whose students were keeping diaries about the difficulties in learning how to program. The other 10 were randomly selected. If necessary, more instructors would be selected, but the analysis showed that the data were repeating, indicating that this number of interviews was enough. Each instructor was identified with an ID comprising a P, followed by a number from 1 to 16. Table I shows the number of years of experience as a professor and gender of each of them.

TABLE II
STUDENTS' DATA.

Age	Total	%
Not inform	1	1%
15 - 24	84	76%
25 - 34	17	15%
35 - 44	6	5%
45 - 54	1	1%
55 - 64	1	1%

TACB	Total	%
0	81	74%
1	17	15%
2	6	5%
3	4	4%
4	1	1%
5	1	1%

The interviews began with a very general question: 'In your view, what are the difficulties faced by students in CS1?' The instructor was then guided to talk about the following introductory subjects: variables, assignment command, input and output commands, arithmetic expressions, relational (<, >, =, ...) and logical (and, or not) expressions, selection structures (if...else), repetition structures (while, for, ...), string manipulation, uni- and bi-dimensional arrays, structured data, functions, and pointers. The subjects were selected by analyzing the university's CS1 program and then compared with the contents covered in the published papers of the area, confirming our selection. Most of the interviews took about half an hour, but some lasted for up to an hour. All interviews were recorded and transcribed for further analysis.

We arranged for students to keep diaries during their studies, keeping us informed about their difficulties as they emerged. We collected data from 6 different classes in 2015 (34 students) and for 6 different classes in 2016 (76 students), from at least 7 distinct departments, yielding a total of 110 student diaries. The classes had on average 53 students, the smallest of which had 29 and the largest 76. All participating students signed a consent giving us permission to read and use the diary data while maintaining their anonymity. The diaries were arranged in shared documents at Google Docs. We asked the students to include code snippets, especially the wrong ones, the approach used to fix the errors, and the doubts they had about learning the topics. To clarify possible misunderstandings about the text written by the students and to encourage them to maintain their diaries, the researchers sometimes posted questions in the diaries. We also asked that the diaries be kept during the whole duration of the introductory programming course, however some students stopped filling in the diary before the course ended. Each student was also identified with an ID, composed by a S, followed by a number from 1 to 110. TABLE II show the total and percentage of students by age range and the number of times they attended the course before (TACB).

B. Data Analysis

For the analysis of the data, we used Grounded Theory (GT) techniques, as described by Strauss and Corbin [29]. During the analysis, concepts, categories, and subcategories emerged. According to Corbin and Strauss [30], "the procedures of grounded theory are designed to develop a well-integrated set of concepts that provide a thorough theoretical explanation of social phenomena under study." The groupings of these concepts into a higher degree of abstraction are called categories [29]. For the analysis, three basic types of coding from the GT are used: open, axial, and selective. Open coding is the process of breaking data down; in axial coding "categories are related to their subcategories, and the relationships tested against data," and in the last one, selective coding, the categories are unified around a core category [30].

The data were read and analyzed, and relevant information was marked with a tag, thus characterizing a concept. Grouping these concepts, 8 categories from interviews and 10 from diaries emerged. Some of them appear in both, for example 'Difficulties with the Topics.' This category has the following seventeen subcategories: Logical Reasoning, Variable, Input/Output, Pseudocode, Expression, Repetition Structure, Selection Structure, Thinking of a Solution, Find Errors, Computer, Programming Language Syntax, Function, Array, X-Dimensional Array, Pointer, String, and Library.

The next step was axial coding. All text marked with codes were read again to identify connections among the subcategories. Connections were created whenever knowledge about one topic became necessary to develop knowledge in another. Fig. 2 to Fig. 12 show these connections. During the axial coding, the subcategories were reorganized. Two subcategories were disregarded: Pseudocode and Library. Since both were cited only once in the data, and they were not linked to any other subcategory. Moreover, the subcategory Exercise was created to separate a subject cited in many other subcategories. The last step was analyzing the connections, understanding what they reveal, and comparing the results with related work found in the literature.

IV. Findings

The aim of this study is to discover connections among programming topics to identify relationships between them that are necessary to develop programming skill. We believe that a shallow learning of one topic may generate consequences for others. Studying the 'Difficulties with the Topics' category, we discovered connections among the topics (subcategories). Each connection indicates that a topic requires another one to aid in the understanding and use of it. All connections are grounded on comments of difficulties in learning how to program, made in the diaries or interviews. In the following, we describe these connections per topic, ordered by the topic most necessary to learn others to the less.

Programming Language Syntax (PLS): C (using Dev-C++ or Code::Blocks, as IDEs) and Python (using IDLE) were the programming languages that were being taught and several references to PLS were made when other topics such as variables and repetition structure were the focus (Fig. 2). This may indicate the importance of offering a solid background on programming language. The difficulty starts with the IDE used, as said P14: "*getting used to the tool we chose for the semester involves a lot of work and demands a lot of energy,*" and S72(C), emphasizing syntax aspects of the language: "*I would like to understand better why they put '# include <stdio.h>', 'int main ()' with two parentheses if we then open*

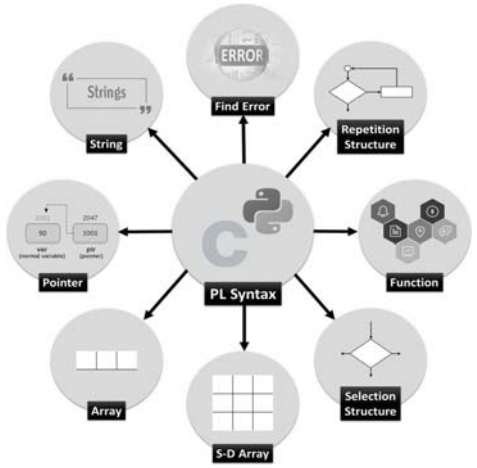

Fig. 2. The relationship between Programming Language Syntax (PLS) and topics that need this knowledge of the syntax.

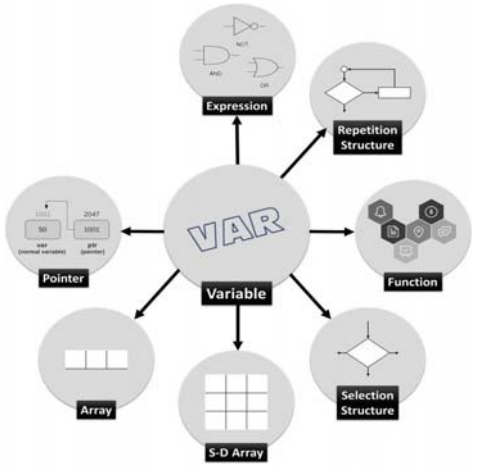

Fig. 3. Diagram showing in which topics the difficulties appear due to lack of knowledge of Variable.

braces, why they chose '\n' as a command to skip line and why they chose to put the letter 'f' in 'printf' and 'scanf'." Syntax errors were the issues frequently commented on by students and instructors: "*...people face problems with syntax...*" - P7. There were issues with PLS related to Repetition Structure (RS) and Functions (Func), such as forgetfulness or missing of braces, "*I made a mistake with the position of the braces in 'while'..., it makes a lot of difference...*" - S41(C). Another common error is with indentation, "*...the biggest difficulty is related to the indentation of the 'if'and 'else' statements...*" - S22, working with Selection Structure (SS) in Python. According to P10, working with Array in C "*brings several misunderstandings that have to do with the language,*" and for P12, working with x-Dimensional Array (xDArray) "*begins to turn into a confusion of indexes and having to think about what is a row, what is a column ... they need to understand that in fact it is not exactly an x-Dimensional Array as we are accustomed, but it is an array of arrays [about how the xDArray is represented in the languages].*" Some instructors avoid explaining Pointer and String when teaching programming with C because the syntax of the language greatly hinders learning.Moreover, students also have difficulty to Find Error (FE) in the code they developed and most of the time the messages that the compiler gives do not help to identify them.

Variable (Var): Students show difficulties learning about variables and applying the knowledge in practice, such as: knowing when it is necessary to create a variable; how to use it, i.e., they confuse name, data, and type of the variable; how much data the variable stores at the same time; among others. Issues about variables were cited in eight other topics when they were being commented on by participants in this study (Fig. 3). Expressions (Expr) are one of them. Students showed that very basic issues about variable were misunderstood, as we can see in the comment made by S3: "'*count = count + 1' What's that? What's the goal of that? Is it adding one to the total amount?*" Moreover, doubts about types were strongly cited when they explained about division results. When working with RS, they highlighted mainly questions about initialization, when and how to change the value by increment or decrement, and difficulty to understand what value was saved in the variable. Working with SS, Array, and Func, doubts about types and their use emerged, like "*... I had some difficulties mainly because it involved char and arrays*" - S34 and "*The idea [that with a] variable you are passing a value and it will be used inside [the function] as a variable*" - P12. Trying to understand about Pointer, "*they [the students] always wanted to get back to using simple variables, to keep everything fixed, everything constant ...*", showing us that there is a confusion between the two topics. Using xDArray, for the students, the variables used to control the positions could not change roles, i.e., if the variable was created to indicate rows, it could never be used to indicate columns, and vice-versa, as occurs in the multiplication of x-dimensional arrays, for example.

Thinking of a Solution (TS): many students have difficulties to develop a solution to solve tasks, independent of language. They frequently understand the theory, or believe they do, but do not know how to apply the knowledge in practice (Fig. 4). An example of this is what P10 explained about Var: "*[the student] uses ten variables where they need three. And there are so many unnecessary variables that he gets confused.*" Working with SS, S3 wrote: "*I had difficulty understanding the use of If, Else.*" Working with Array, S61 exposed that she/he was not sure "*whether it's necessary to use the concept of array.*" Sometimes, the students complete the exercises only with help from somebody, as S30 explained when working with RS: "*[when] 'While' was introduced, I had difficulty working with the last list [of exercises], I was able to do it with the help of a classmate, even though during class I felt that I had understood perfectly how it worked.*" Another difficulty faced by students was FE in the solution,

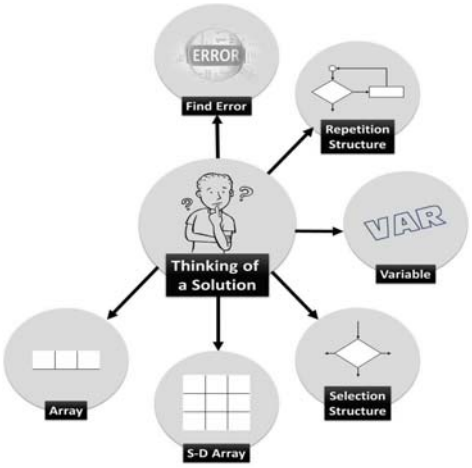

Fig. 4. Diagram showing in which topics the difficulties appear due to lack of skill in Thinking of a Solution.

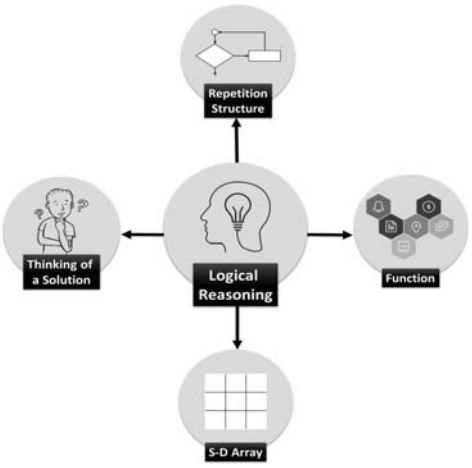

Fig. 5. Diagram showing the connection of Logical Reasoning with other topics.

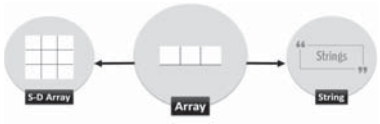

Fig. 6. Diagram showing in which topics the difficulties appear due to lack of knowledge of Array.

S34 stated that she/he changed the solution and that anyway she/he still had not understood what was wrong. And the last one that included comments on developmental difficulties was the topic xDArray, "*Although I drew, talked about how they should look at it, there were some [students] who could not produce algorithms, working with the xDArray*" - P7.

Logical Reasoning (LR): This topic is considered one of the most important for learning programming, as confirmed by P15: "*I believe that the first point is logical reasoning*" and P12 - "*logical reasoning is maybe more important than each topic of the course.*" Many reports show the lack of this skill, as said by instructor P11 - "*Any simple algorithm is very hard in their head because they do not have [it] [...] it is a completely different way of thinking from what they are accustomed to*" and confirmed by students with many comments, such as: "*I have a hard time turning a problem, often simple to solve with paper and pen, into a program*" - S19 and "*I had difficulties in the logic of assembling the algorithm ...* " - S1. Instructors work in classes to improve this skill: "*The focus of the classes is usually the exercise of algorithms, to develop students' ability to think of solutions*" - S71 but even so the students continue to have difficulty to TS to develop code, "*my ability to pick up the theory and apply it to the exercises on my own is negligible,*" showing the necessity to work hard to improve logical reasoning in students (Fig. 5). With the topics RS and Func, students have difficulty to realize the correct order of actions and commands to achieve their aims, "*One difficulty I feel is how to order actions, for example: should I finish all the functions and then test if it worked out, or is it better to test before, but how to do it? Since I do not know how to test without the program actually written*" - S2. Another topic that required LR is xDArray, where students did not know what to do, as commented by S50: "*I spent all day trying to make the third EP (homework) after studying a lot about x-Dimensional arrays, but I only managed to spend more than six hours staring at the monitor feeling my brain fry without knowing how to start doing what the EP asked for.*"

Array: Some difficulties faced by students working with arrays are manipulation of indexes and lack of prior knowledge, such as in math, "*If the person [student] does not have the concept of array and x-Dimensional Array, it becomes difficult to understand ...*" - P5. Two other topics needed knowledge of array to be learned (Fig. 6). The first was xDArray, "*X-Dimensional Arrays start to turn into a maze of indexes and having to think of what is a row, what is a column... they have to understand that in fact it is not exactly an x-dimensional array like we are used to but a vector of vectors, which is something that is a bit difficult as well*" - P12, and the other one was String, such as commented by P15 - "*Difficulties using strings, ...when I look at a string I have the characters, each occupying a position, seeing a string as if it were an array with a limit and an end marker, in the case of the C language.*"

Pointer: For P9, when teaching pointers, "*the trouble starts to get bigger,*" and some instructors avoid explaining this topic because of the difficulty that students have in understanding it. Students do not like it, as shown by the outbreak of S42 when working with Func: "*I hate functions... passing parameters by reference...welcome to the hell of pointers! * ** ** && * & * !! Ugh.*" And another topic that required pointer knowledge was Array (Fig. 7). Understanding an array as a pointer is important when passing an array to a function, for example.

Expression (Expr): SS and RS are the two topics that needed the knowledge of Expression to be learned (Fig. 8). For both SS and RS, it is necessary to develop the condition

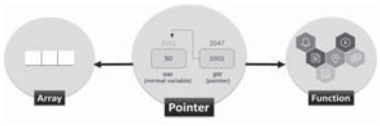

Fig. 7. Diagram showing in which topics the difficulties appear due to lack of knowledge of Pointer.

Fig. 8. Diagram showing in which topics the difficulties appear due to lack of knowledge of Expression.

Fig. 10. Diagram showing in which topics the difficulties appear due to lack of knowledge of Input and Output Functions.

Fig. 11. The relationship between Exercise and Thinking of a Solution.

that controls the structure. In this task, the use of expressions is necessary and, according to P11, "*...there is one thing that exists in natural language and that normally does not exist in programming languages, it is something like 'x > 3 and < 5' and in computing I cannot write it like this, I have to write 'x > 3 and x < 5', so they write it wrong, they write 'x > 3 and < 5'*" and S24 confirmed showing that she / he tried to write the condition in that way: "*If a > b > c:*" (Python).

Repetition Structure (RS): Creating the stopping condition, manipulating the control variable and understanding and creating nested repetitions are only some of the problems faced by students learning repetition structures. If the students do not learn this topic correctly, they could also have problems with xDArray (Fig. 9), which uses nested repetitions for manipulation, as commented by P13 - "*I do not see problems in them seeing or understanding array or x-dimensional array, I see problems in manipulating them. Using a nested repetition, for example?*"

Input/Output (IO): This topic was required to learn and work with RS (Fig. 10). Students had problems to define what data were needed to work in the program and how to get them. To work with the data, the students did not know if it would be necessary to use the input command to insert the data in the program. Another issue about input/output command was that some students thought that the users could see the results without showing them using the output command. And finally, the students had difficulty setting the correct position of the input / output commands in the code: "*Python is printing the whole sequence, instead of just printing the final output, what do you suggest for me to solve that? [asking the researcher]*" - S22.

Exercise: the description of the exercises needs to be well written. The topic was required by TS (Fig. 11). To use the theory to develop the code, it is necessary that the student can understand correctly what the task is: "*I tried to read the EP [homework], but I could understand absolutely nothing of what it was asking for*" - S25.

Device Operation: P8 believes that "*the difficulty is in understanding how the computer solves the problems that are very trivial to them and they come up with ways of solving them that they already know, that they are used to, and cannot think as the machine thinks. Not 'thinks', solves. So that is the great difficulty*". This topic was required by TS because "*knowing how things work inside the computer is very useful because from this you can do schematics that help you think better about the situation and how to proceed to reach a response program*" - S35 (Fig. 12).

X-Dimensional Array (xDArray), **Function (Func)**, **Find Errors (FE)**, **Selection Structure (SS)**, and **String:** Although these five topics are not necessary to aid in the learning of others, they are topics that require a lot of prior knowledge so that the student can learn them. As an example, students can Find Error only if they have the knowledge of the topic Programming Language Syntax. Another example is learning to develop and use functions, which uses prior knowledge of variables, pointers, and so on.

The results showed the connections between topics needed to learn how to program. These connections show the existence of prerequisites between them. Many difficulties were presented in the literature, as seen in the Related Work section. In analyzing the data, we found that many of the reports about the difficulties were not related to the topic being studied, but rather to a lack of knowledge of other topics. This illustrates the importance of knowing these connections well and knowing what knowledge is necessary so that something

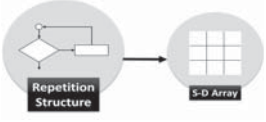

Fig. 9. Diagram showing in which topics the difficulties appear due to lack of knowledge of Repetition Structure.

Fig. 12. The relationship between Device Operation and Thinking of a Solution.

new can be learned.

V. Discussion

Our theory shows and catalogs the relationships between topics that are necessary to develop programming skills. To illustrate, let us take Repetition Structures (loop), 'While' and 'For', as an example. To learn or teach Repetition Structure, it is necessary to have knowledge about other topics, such as expressions to build the stopping condition and maybe to change the value of the control variable, which means it is also necessary to know about variables, attribution, etc.

We can perceive how essential this interconnection is when our participants comment about difficulties regarding topics that is not the focus at the moment. Students and instructors highlight the importance of knowledge about Variables, for example, and this is one of the topics considered difficult by Sevella and Lee [25]. Function, another topic cited by them, is also connected to Variables, as well as repetition, list (array), pointer and passing parameters (Function), all cited as the most difficult topics by Mhashi and Alakeel [23]. Also, the influence of one topic in the understanding of other topics may be seen in some sentences like these: "*The problem was in the second 'FOR'...*" - S18 - showing the necessity of knowledge of Repetition Structure while working with xDArray, and "*I could not do it, it gave an error about the variable when using the pointers*" - S41 while she/he was learning function, showing the importance of the knowledge of pointers to this subject.

These and other topics need to be used in practice, and for that there are three important Topics: Thinking of a Solution, Programming Language Syntax and Logical Reasoning. Although applying the topic needs the programming language, and consequently knowledge of its syntax, to develop the programs, these two topics are not directly linked in the graph. However, most of the topics are linked to both. Also, to apply the topic in practice, it is also necessary that logical reasoning is developed for this area of application, thus enabling ideas on how to proceed to reach the goal, as confirmed by Garner and colleagues [7]. For Thinking of a Solution in practice, beyond Logical Reasoning, two other topics were required. The first was Exercise, where the importance of a good description was highlighted. Without understanding and knowing what the task is, it is unlikely that it will be solved. About the importance of good and clear exercises was also discussed by Giraffa and Moura [31] and Gomes and colleagues [3]. The second was Device Operation, also cited by Milne and Rowe [4], emphasizing the importance of knowing how it works in general. And the last topic directly connected with PLS and TC is Find Error, a topic also cited by Gomes and Mendes [27]. Finding syntax errors and the errors in the use of topics were often brought up as difficult issues by both students and instructors.

A. Threats to Validity

The interviews were conducted with instructors from a single department (Computer Science) but who teach in different schools (Physics, Chemistry, Engineering etc.) of the University; consequently, the diaries from students are also from these various faculties. This may raise some doubts as to the veracity of the results in other contexts. Would the results be the same if the data were collected at another university, with other technologies being used and cultural aspects that differ from those in the environment where the data were collected? To answer these issues, some parallel studies could be taken, such as a survey could be created to confirm the results obtained, which should be applied in different universities located in different regions, preferably in different countries and continents. Our study did not take into account the potential cognitive effects of asking students to keep a diary during their participation in the course, perhaps this may have caused some bias in the results obtained.

VI. Conclusion

Learning to program is not an easy task and this paper goes a step ahead in understanding why. In this paper, we show a discussion about the importance of learning deeply each topic because of the interconnections among them. Our view is grounded in 16 interviews made with instructors and 110 diaries filled by students during their studies.

This study showed that misunderstanding one topic may be caused by the difficulty that appears in another. This can lead us to evaluate the importance of paying attention to teaching-learning of each topic, distributing the time and attention given, so that they can be worked out avoiding as soon as possible that difficulties are propagated between them.

We contribute with a theory that describes the connections among topics of learning how to program. These connections show the importance of keeping attention in the sequence of topics taught in class. A poorly taught or learned topic can create difficulties in learning others. Better knowledge about these connections can give course coordinators and instructors the opportunity to change the way of teaching, helping students to avoid some of the difficulties faced on learning how to program.

Acknowledgment

We thank all the instructors in our study for their time to receive us to talk about teaching programming. We also thank the students that filled the diaries, providing a rich material for our research.

References

[1] I. Drosos, P. J. Guo, and C. Parnin, "Happyface: Identifying and predicting frustrating obstacles for learning programming at scale," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 171–179, IEEE, 2017.

[2] K. Ala-Mutka, "Problems in learning and teaching programming-a literature study for developing visualizations in the codewitz-minerva project," *Codewitz needs analysis*, vol. 20, 2004.

[3] 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*, pp. 1–8, IEEE, 2014.

[4] I. Milne and G. Rowe, "Difficulties in learning and teaching programmingviews of students and tutors," *Education and Information technologies*, vol. 7, no. 1, pp. 55–66, 2002.

[5] T. Jenkins, "On the difficulty of learning to program," in *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, vol. 4, pp. 53–58, Citeseer, 2002.

[6] M. Hristova, A. Misra, M. Rutter, and R. Mercuri, "Identifying and correcting java programming errors for introductory computer science students," *ACM SIGCSE Bulletin*, vol. 35, no. 1, pp. 153–156, 2003.

[7] S. Garner, P. Haden, and A. Robins, "My program is correct but it doesn't run: a preliminary investigation of novice programmers' problems," in *Proceedings of the 7th Australasian conference on Computing education-Volume 42*, pp. 173–180, Australian Computer Society, Inc., 2005.

[8] D. Weintrop, "Blocks, text, and the space between: The role of representations in novice programming environments," in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 301–302, IEEE, 2015.

[9] M. J. Lee, "Gidget: An online debugging game for learning and engagement in computing education," in *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 193–194, IEEE, 2014.

[10] M. Ichinco and C. Kelleher, "Towards block code examples that help young novices notice critical elements," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 335–336, IEEE, 2017.

[11] I. Bergström and A. F. Blackwell, "The practices of programming," in *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 190–198, IEEE, 2016.

[12] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming," *AcM SIGcSE Bulletin*, vol. 39, no. 2, pp. 32–36, 2007.

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

[14] Y. Bosse and M. A. Gerosa, "Why is programming so difficult to learn?: Patterns of difficulties related to programming learning mid-stage," *ACM SIGSOFT Software Engineering Notes*, vol. 41, no. 6, pp. 1–6, 2017.

[15] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," *Acm Sigcse Bulletin*, vol. 37, no. 3, pp. 14–18, 2005.

[16] G. Fischer and E. Giaccardi, "Meta-design: A framework for the future of end-user development," in *End user development*, pp. 427–457, Springer, 2006.

[17] P.-H. Tan, C.-Y. Ting, and S.-W. Ling, "Learning difficulties in programming courses: undergraduates' perspective and perception," in *2009 International Conference on Computer Technology and Development*, vol. 1, pp. 42–46, IEEE, 2009.

[18] C. Jiménez and J. Villalobos, "Learning/teaching a computer programming course," *Analysis of State-ofthe-Art Solutions for Personalised Learning Support*, p. 3, 2010.

[19] F. Alonso, G. L. Gómez, J. M. Font, and D. Manrique, "Learner satisfaction when applying an instructional model in e-learning-an experimental study.," in *CSEDU (1)*, pp. 141–146, 2010.

[20] M. J. Van Gorp and S. Grissom, "An empirical evaluation of using constructive classroom activities to teach introductory programming," *Computer Science Education*, vol. 11, no. 3, pp. 247–260, 2001.

[21] J. Hamer, Q. Cutts, J. Jackova, A. Luxton-Reilly, R. McCartney, H. Purchase, C. Riedesel, M. Saeli, K. Sanders, and J. Sheard, "Contributing student pedagogy," *ACM SIGCSE Bulletin*, vol. 40, no. 4, pp. 194–212, 2008.

[22] J. Lang, G. C. Nugent, A. Samal, and L.-K. Soh, "Implementing cs1 with embedded instructional research design in laboratories," *IEEE Transactions on Education*, vol. 49, no. 1, pp. 157–165, 2006.

[23] M. M. Mhashi and A. Alakeel, "Difficulties facing students in learning computer programming skills at tabuk university," in *Proceedings of the 12th International Conference on Education and Educational Technology (EDU13), Iwate, Japan*, pp. 15–24, 2013.

[24] M. Piteira and C. Costa, "Learning computer programming: study of difficulties in learning programming," in *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, pp. 75–80, ACM, 2013.

[25] P. K. Sevella, Y. Lee, and J. Yang, *Determining the barriers faced by novice programmers*. PhD thesis, Texas A & M University-Kingsville, 2013.

[26] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[27] A. Gomes and A. J. Mendes, "Learning to program-difficulties and solutions," in *International Conference on Engineering Education–ICEE*, vol. 2007, 2007.

[28] J. W. Creswell and J. D. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.

[29] A. Strauss and J. Corbin, "Basics of qualitative research: Techniques and procedures for developing grounded theory 2015; 4."

[30] J. M. Corbin and A. Strauss, "Grounded theory research: Procedures, canons, and evaluative criteria," *Qualitative sociology*, vol. 13, no. 1, pp. 3–21, 1990.

[31] M. M. Giraffa and M. da costa Mora, "Evasão na disciplina de algoritmo e programação: um estudo a partir dos fatores intervenientes na perspectiva do aluno," in *Congresos CLABES*, 2013.