Training of highly qualified specialists in the field of software development: problems and solutions in higher educational institutions of Ecuador

Valeria Esthefania*, *Quiroz* Cabascango, and *Diana Esmeralda* Molina Calvopiña

PUCE TEC, Pontifical Catholic University of Ecuador, Quito, Ecuador

Abstract. The article presented analyses the difficulties faced by a teacher in the Software Development program and the tools they can use to improve the effectiveness of the teaching process. The author separates algorithmic thinking from software development technology. It is noted that the problem of teaching how to develop software is associated with the understanding of the subject area. The second part of the article examines the teacher's options when choosing ways to improve the quality of education in the Software Development career. The question of choosing a programming language is analyzed. And, programming environments that have little influence on the learning process and outcome are considered. The authors focus on the interaction between "teacher and student" and the importance of a technique in teaching such as automating the verification of tasks completed by students.

1 Introduction

According to many experts, the growing demand for Software Development technologists in Ecuador can last up to 10 years or more. To increase market saturation with technologists in this branch, Higher Education Institutions have developed a series of measures. In particular, they have proposed creating technical training programs in Software Development. For its part, the Government of Ecuador has implemented the "TEC and Fortalécete 2023 Scholarship Programs", in order for students to take software development as a second career or a complementary career to meet the demand of the country and region [1, 23]. This means that there are similar training programs throughout the country, both in universities and institutes. By 2023, the number of budgetary places in universities with specialties related to technology in Software Development has increased significantly.

The training of technologists in the field of Software Development also occurs in large companies that themselves provide training in the disciplines they need.

Of course, the training of technologists in the field of Software Development are important tools that produce certain results. But it must be taken into account that training can be presented in the form of a black box, where at the entrance are those who will be

* Corresponding author: valeritae111@mail.ru

trained and at the exit those who are already trained. Both the quantity and quality of trained people may not always meet the required requirements. What and how to teach in the field of programming is becoming one of the most important problems in the training of modern technologists in the field of information technologies [2]. For future professionals, this poses a special challenge: mastering technology is not always the main learning issue. Many times the difficulty lies in the fact that the student does not fully understand the logic of the computer (executor) during the execution of the program, and in this case writing his own programs becomes an insurmountable obstacle for him.

In this article, we will look at some of the difficulties of teaching Software Development in higher education. Characteristics of students' perception of the programming process, the program execution process, as well as the software debugging and testing processes [3]. We will discuss possible approaches to improve the effectiveness of teaching programming.

2 Learning difficulties for a software developer student

Training in software development is always carried out in two directions: algorithmic thinking and technology. Technology refers to a whole set of tools: a development environment, a programming language, libraries and interaction with the operating system, debugging tools and code analysis. By algorithmic thinking we mean the ability to build a sequence of actions, the implementation of which will lead to a certain goal. Of course, people constantly build such sequences throughout our lives to achieve goals. The thing, however, is that the average person builds life algorithms based on logic, intuition, emotional choice, etc [4].

A software development student is faced with the fact that he needs to change the way he creates or edits such algorithms. First of all, in this case the algorithm must unambiguously lead to the achievement of the goal. Second, the algorithm is done by a computer, not a person. These problems are unknown to professional programmers, but for a student they can become a serious obstacle [5].

It is important to see the boundary between an algorithm and a programming language. Let's give two examples [6].

Example 1: Assigning a value to a variable is an element of the algorithm. But in different languages such an assignment can be done in different ways, even in several ways. For example, in C language, and a = 50; and a = b = 50; and int a = 50; gives the same result, but everything is an assignment to a variable [7].

Example 2. Searching for a simple element in an array in a programming language can be implemented in different ways: various loops, search functions, methods of the array object.

For a professional developer, this element is revealed only at the stage of writing the program. This example is interesting because as programming skills develop, certain algorithms begin to “collapse” for the student, becoming a simple element that “unfolds” only when the program is written. For a beginner developer, the search algorithm itself may present some difficulty, not to mention the possibility of writing it within a given programming language in different ways [8].

The analysis in this article identifies three types of typical errors made by software development students, directly related to understanding how a computer works.

Concurrency errors. The student assumes that the computer perceives the program as a whole, regardless of which line of the program is currently running [9,10].

Errors of intentionality (direction). The student draws erroneous conclusions about the future actions of the computer based not on the entire program, but only on a fragment of it. The student does not correctly interpret the meaning of the rest of the program.

Errors of egocentrism. The student mistakenly believes that the computer is somehow capable of "understanding" the objectives of the program. This is usually expressed in the fact that the student skips some important elements (conditions, cycles), believing that the computer itself will reproduce these elements.

Software development encompasses several types of activities: writing a program, testing, and debugging. Testing, as a process of checking whether a program meets a set of certain requirements, is often reduced in universities to an automatic testing of the program by comparing it with predetermined tests. This approach, which has been increasingly used in recent years, is undoubtedly an important method to improve the quality of programming education. But this misses an important link in learning: the student is not involved in developing the test material for the task. At the same time, if the student has access to a set of tests, then a gap arises between the conditions of the task and the program that it is developing [11,12].

The program debugging mentioned above is closely related to understanding the program text. The main task of debugging is to locate errors in the program. Very often, novice developers solve the problem assigned to them, which is directly related to programming, but find themselves absolutely helpless before the problem of finding and understanding bugs.

Some authors maintain that even experienced programmers often have certain deficiencies in debugging [13,14]. That said, those who are good at debugging tend to be good programmers. This phenomenon can only be explained by the fact that writing a program and understanding the program text do not always go in parallel. Why is this happening? At least four reasons can be given:

1. When writing a program, a developer usually uses ready-made templates. It does not matter whether he inserts a ready-made fragment into his program or whether this template is, in fact, taken from his memory.
2. This may seem surprising, but very often a developer uses intuition when writing, based on the experience he already has in the field of programming. The result of this approach essentially boils down to the result of point 1.
3. If the program is large enough, then the meanings of individual lines, fragments and structures may, after a while, not be entirely clear to the programmer, since they were integrated into some general logic that was created in the process program writing. Some components of this logic are lost after some time.
4. Programming languages (their implementations) have a large number of features that can only be extracted from a good reference manual. None of the programmers (and especially the students) checks every element (operator, function) that he uses with the reference book. The error may arise simply due to the fact that some feature of the programming language was omitted.

Another important problem that arises in the teaching of the software development program in higher education is the inequality in the training of students. Recently, schoolchildren receive programming training as part of their secondary education or additional training courses. These students, far from being the best students, are significantly superior to their peers in programming subjects. This is an interesting phenomenon that is occurring with increasing frequency in our higher education institutions and apparently requires a new approach.

An important problem is raised in this article. The question arises, what is associated with a person's ability to master a discipline such as software development? In particular, they talk about a weak connection with general education and the level of intellectual development. Authors of the works consulted believe that the predisposition (or lack of predisposition) to activities such as programming is inherent in some people from the beginning. This controversial statement, however, does not eliminate the issue of career guidance. In particular, additional school education, project activities in schools, a system of competitions

and the possibility of conducting a preliminary selection of schoolchildren for future activities in the field of software development [15].

There is a group of problems associated with mastering specific programming elements. For example, variables, assignments, conditionals, select statements, loops, complex structures.

There are large and complex sections in programming, such as window applications, network programming, web programming, mobile device programming, microcontroller programming [16]. The problem here lies precisely in algorithmic thinking and understanding how the program works. In other words, the student does not always understand the topic area she addresses when solving a problem. Often the problem for a modern student stems from the ability to read and understand texts.

3 Methodology and possible solutions

In this research, qualitative methodologies were used focused on obtaining information on the experiences and perceptions of the participants that are of interest to the research. Below we will discuss approaches that can be used to improve the effectiveness of teaching programming.

3.1 Programming languages

Does language choice affect mastery of basic programming concepts? This question is discussed very often. But upon closer examination, it turns out that the choice of programming language is linked not so much to the ability to develop and write algorithms, but to other aspects. Let's list them.

a. There are simple and difficult programming languages to learn. For programming beginners, you should choose a language that is easy to learn. Very often lately, especially in school education, Python is chosen as such a language. In fact, there are simpler programming languages created specifically for learning. [17,18] But the question always arises: is learning to program just about learning a language? A writing classification algorithm (any) for all languages will have approximately the same difficulty in understanding the student.
b. Another argument used to justify the choice of a programming language is related to the strictness of the language rules. In particular, we are talking about strict typing of variables. Among some experts in the field of programming there is an opinion that the first language should be a simple language and at the same time subject to strict rules, such as strict typing or a strict program structure. Among these languages is in particular Pascal, which until recently was very popular in school computer courses.
c. This argument is also sometimes found, especially in Internet articles. It is not about using special educational languages, but about immediately choosing an industrial programming language, that is, the language in which the software is created. And in this case, the Pascal language is viewed with some caution, since according to many (although it is generally a wrong opinion), it has not been used in real development for a long time.
d. This argument can be seen in various Internet articles about programming. This reasoning is based on the popularity of programming languages among employers. There are special sites that collect information about the popularity of programming languages.

Concluding the question of choosing a programming language for training, we will refer to the opinions of the leaders of software development groups, the essence of which boils

down to the fact that knowledge of a specific programming language is not yet the main professional competence. Therefore, a programming language is unlikely to facilitate the task of developing algorithmic thinking in students, but it can develop in the student interest and certain skills related to writing professional programs.

3.2 Programming environment

How important is the programming environment in learning? And here, as in the case of programming languages, it is often argued that the most developed environment is needed, with the possibility of intelligent suggestions, running a program directly from the editor and all the debugging tools. Let's express a series of considerations:

1. An environment that frees the learner from some of the work of writing language items by suggesting possible options first partially creates a gap in professional skills; Second, it frees the student from responsibility for what is written, which can actually complicate the program debugging process.
2. The development environment creates a certain barrier between the programmer and the execution environment. After all, the real execution environment is the operating system. We will return to this quote below when we talk about student motivation.

In our opinion, working in programming environments such as Eclips, PyCharm, NetBeans, etc. It is interesting from the point of view of familiarizing future specialists with professional development tools with which they may have to work in the future. This is where the benefit of choosing a development environment essentially ends [19].

Since we are talking about basic programming training, we will not stop at a tool as well-known as visual programming. This tool is undoubtedly good for creating windowed applications, but, in essence, it is based on basic algorithmic programming skills, as well as such an important paradigm as object-oriented programming [20].

3.3 Motivation

Motivation has always been considered the most important component in learning to program or develop software. Let's consider several of its important components (motives) in relation to teaching programming [21].

1. Probably the most important reason for learning to program is the desire to become a professional programmer. A programmer (software developer) in our country is quite well off financially and, at the very least, can always find a job [22]. And if we take into account the digitalization processes that are being developed in Ecuador, the prospects for future programmers increase significantly. Additionally, there are prospects of working in international corporations, which is an important factor for some students.
2. Let us also highlight a reason, which I would separate from the reason indicated above. An important factor is belonging to the community of professional programmers, as a certain segment of society that is not directly related to salary level.
3. This type of motivation occurs quite frequently, but is not predominant. You can call it: pleasure of the programming process. In the article we consider only activities related to the programming process, but in reality this motive is associated with a more general motive: the pleasure of intellectual activity. One way to develop such motivation is to complete tasks related to elements of the research. For example, a student must write a program when she is evidently unaware of some of the technologies she will need to use.

4. This reason must be separated from reason number 3. We would call it “expectation of the new.” It always happens to some students when they start learning a new programming language, new technology or paradigm. And in this sense, it seems to us, we should not focus on a programming language or a paradigm. A golden mean is needed. If the basic programming language is, for example, Pascal (Free Pascal), then you should not try to "squeeze" all possible results: graphics, network programming, etc. It is quite possible to include individual topics in course projects [24].

It is clear that for a student programming is not the process of creating software, but only the fulfillment of the teacher's tasks. Here is an approach for students beginning to program:

a. The program is written in a simple text editor.
b. Compiled or interpreted directly using operating system commands.
c. Testing of written programs is carried out directly in the operating system (normal terminal or terminal in Meadnight Commander). The program receives input data in the input stream or as command line parameters.

Students will instantly grasp the essence of this technology, regardless of their initial training. In addition, understanding the structure of the file system is not at all superfluous for future professionals in the field of information technologies.

3.4 Paradigms in software development

Regarding the problem of teaching programming, one cannot ignore an issue as important as programming paradigms. When teaching programming, it is not necessary to start from the historical stages of programming development. When teaching, it is convenient to distinguish three main paradigms: procedural programming, modular programming, object-oriented programming. It is not necessary to pay special attention to other paradigms: functional programming, declarative programming, structured programming, etc. in the teaching process. The only thing here, in our opinion, important from the point of view of increasing the effectiveness of teaching is the sequence of presentation of the material [25]. Naturally, starting with an object-oriented approach would not be the right solution. The sequence presented above is precisely the most optimal, since the indicated paradigms are included in the following as subsets.

3.5 Teacher and student team

The teacher and the student are a key construct in the entire higher education system. The interaction in this "couple" has been changing very rapidly lately. But we will not address the educational process at all. We will only point out what is directly related to teaching software development.

In the field of programming, the following interesting phenomenon draws special attention. Some students in groups are significantly separated from the general mass, even if the majority of the group successfully masters the materials of the discipline. How to respond to this phenomenon: transfer particularly successful students to individual programs, use them as teaching assistants, or some other option? We don't know the answer to this question. However, this phenomenon exists and undoubtedly affects the entire teaching process.

One of the important innovations, which is already used in many universities, is the transition to automatic verification in terms of solving programming problems. There is suitable software for this. Of course, not all programs are suitable for automatic verification. But it makes sense to try programs to practice basic programming skills, including the important part of algorithmic thinking. To do this, the tasks themselves must be formulated

in a certain way, according to the principle: input - output information flow. What can be achieved with this approach?

1. It frees the teacher from the need to arduously review students' programs to pay more attention to individual work, preparation of educational materials, etc.
2. Teach students to write programs that exactly satisfy the given conditions.
3. Break the chain of dependence of the teacher and the student on each other in terms of checking most of the tasks.

4 Conclusions

We examine several problems associated with teaching programming in higher education:

1. The development of algorithmic thinking is fundamental to software construction and the code debugging process.
2. The gap between the programming process and the understanding of how to solve requirements through code (programming language).
3. Groups of students with heterogeneous technical skills make the learning process difficult.
4. Understanding of the subject area.

Elements of teaching were identified whose improvement could improve the quality of teaching:

1. Appropriately select the development environment and programming language according to the technical level of the students so that the tools in the environment serve to support the learning process.
2. Motivation is a determining factor when acquiring knowledge, which is why emphasis must be placed on the types of motivation and ways to influence them within the technical approach.
3. The interaction between teacher and student is essential for the educational environment to strengthen the technical skills required for software development.

References

1. Government of the Republic of Ecuador 2023 The Government presents the Tec 2023 Scholarship Program for 20 000 young people (2023). URL: https://www.educacionsuperior.gob.ec/el-gobierno-presenta-el-programa-becas-tec-2023-para-20-000-jovenes/ (accessed: 30.10.2023)
2. N.O. Kuligina, *Methods of teaching and assessing students' knowledge of programming at the Department of AIS DPI NSTU* Collection of proceedings of the conference of the Nizhny Novgorod State Technical University named after. R.E. Alekseeva **1**, 198-202 (2014)
3. M.A. Arbuzov, Journal of Mining Institute **155(1)**, 263-265 (2003). https://pmi.spmi.ru/pmi/article/view/9125
4. E.I. Pryakhin, E.V. Larionova, E.A. Zakharenko, Journal of Mining Institute **209**, 234-238 (2014). https://pmi.spmi.ru/pmi/article/view/5317
5. V.N. Balandin, I.V. Menshikov, M.Y. Bryn, Y.G. Firsov, S.L. Shtern, Journal of Mining Institute **204**, 24-28 (2013). https://pmi.spmi.ru/pmi/article/view/5544
6. H. Abelson, *Structure and interpretation of computer programs* (Moscow, KDU, 2012), p. 608
7. V.Yu. Bazhin, T.R. Kosovtseva, A.Z. Muzipov, Non-ferrous metals **4**, 31-37 (2023). https://www.doi.org/10.17580/tsm.2023.04.04

8. V.E. Quiroz Cabascango, V.Yu. Bazhin, S.A. Martynov et al., Metallurgist **66**, 104-116 (2022). https://www.doi.org/10.1007/s11015-022-01304-3
9. M. Konecki, *Problems in programming education and means of their improvement* (Daaam international scientific book, 2014), **37** 459-470. https://daaam.info/Downloads/Pdfs/science_books_pdfs/2014/Sc_Book_2014-037.pdf
10. V.Yu. Bazhin, O.N. Masko, S.A. Martynov, Metally **4**, 53-60 (2023). https://www.doi.org/10.17580/tsm.2023.04.07
11. E. Samylovskaya, A. Makhovikov, A. Lutonin, D. Medvedev, R. Kudryavtseva, Resources **11** (2022). https://www.doi.org/10.3390/resources11030029
12. P.A. Petrov, A.K. Shestakov, M.Yu. Nikolaev, Tsvetnye Metally **4**, 45-53 (2023). https://www.doi.org/10.17580/tsm.2023.04.06
13. H.H. Nguyen, V.Y. Bazhin, Metallurgist **67**, 41-50 (2023). https://www.doi.org/10.1007/s11015-023-01487-3
14. S. Dehnadi, R. Bornat, The camel has two humps (working title) School of Computing, Middlesex University (2006). https://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf (accessed: 30.10.2023)
15. V.V. Parchenkova, *Algorithmic thinking and ways of its development* Youth and the XXI century: materials of the VIII International Youth Scientific Conf. Kursk **21**, 191-192 (2018)
16. A.V. Pavolotsky, Quality Innovation Education **12**, (2015). https://publications.hse.ru/pubs/share/folder/r53cdc9c79/174489097.pdf (accessed: 15/10/2023)
17. A.N. Bobrov, Young scientist **24(104)**, 61-64 (2015)
18. M. Fatih Yiğit, M. Baser, Participatory Educational Research (PER) Special Issue **2,** 27-33 (2015). https://www.perjournal.com/archieve/spi_15_2/4_per_15_spi_2_4_Page_27_34.pdf (accessed: 1.11.2023).
19. V.Yu. Pirogov, Pedagogy and psychology **6(6)**, (2018). https://mir-nauki.com/PDF/67PDMN618.pdf (accessed: 15.10.2023)
20. A. Gomes, A.J. Mendes, *Learning to program — difficulties and solutions*, International Conference on Engineering Education (2007). http://icee2007.dei.uc.pt/proceedings/papers/411.pdf (accessed: 30.10.2023)
21. L.Y. Zobnina, Journal of Mining Institute **163**, 120-122 (2005). https://pmi.spmi.ru/pmi/article/view/8364
22. E.B. Mazakov, Journal of Mining Institute **208**, 256 (2014). https://pmi.spmi.ru/pmi/article/view/5365
23. A.P. Mosquera Urbano, T.V. Ponomarenko, *Improving energy efficiency in the Republic of Ecuador*, International Multidisciplinary Scientific GeoConference : SGEM **20**, 285-292 (2020). https://www.doi.org/10.5593/sgem2020/5.2/s21.035
24. G.B. Pospehov et al., IOP Conference Series: Earth and Environmental Science **1212**, 012029 (2023). https://www.doi.org/10.1088/1755-1315/1212/1/012029
25. V.P. Zubov, L. Yunpeng, MIAB. Mining Inf. Anal. Bull. **7**, 37-51 (2023). https://www.doi.org/10.25018/0236_1493_2023_7_0_37