Logic Programming: Can the learning and teaching process be improved apart from standards adopted by most professors and established in academic books?

Ricardo Luiz B. L. Campos

Serviço Federal de Processamento de Dados – SERPRO
SGAN 601 Módulo G – SUPDE/DEBSA/DE1BD
70743-040 – Brasília/DF – Brasil
rluizcampos@yahoo.com.br

Abstract. Logic programming is one of the most important subjects for students pursuing a degree in Information Technology and Communication (ITC) at higher education institutions. Teaching logic programming is considered one of the seven greatest challenges of computer science education in this century due to the high rate of failure among students, and because it directly influences how students will perform in other related subjects for the remainder of their program. Therefore, researching new strategies such as the ERM2C methodology is essential for improving education in the area. The methodology proposes that the process be developed in two stages: the first for reading/understanding any algorithm, and the second for building total or partial logic to solve problems. The methodology is still in the validation process, but the results obtained so far have been considered satisfactory.

Key words: logic programming methodology for developing logic programming.

1 INTRODUCTION

One of the most important subjects for students pursuing a degree in Information Technology and Communication (ITC) is logic programming [1]. It is also the subject that has the highest rate of failure in Brazil and in the rest of the world [1][2][3][4][5][6][7][8]. For this reason, it's one of the seven greatest challenges in computer science education in this century (McGettrick *apud* [9]).

The explanation for this statement is that the knowledge students acquire in this subject will directly influence their performance in other related subjects throughout the remainder of their program [1]. Therefore, studies to identify the reasons behind the high failure rate are essential, as are strategies for improving the learning/teaching process [2][3].

Interviews with local university students identified that the vast majority has difficulty creating algorithms for solving sometimes even basic problems. However, interviews with faculty members of these same institutions indicate that building algorithms are the problem-solving strategies used in the classroom and in textbooks.

If students have trouble solving problems using this method taught by most professors, there is no guarantee that students will learn and consequently be approved in this subject?

An alternative can be the ERM2C methodology, which combines the traditional method adopted by most professors and what have been identified as the best strategies being used by professors with lower failure rates.

The ERM2C methodology was tested during 5 semesters: 4 with volunteer students and one at a private programming logic course. The result, despite not being conclusive, can be considered satisfactory.

This article presents the ERM2C methodology and is divided in 5 sections: 1. Introduction, 2. the problem with the learning/teaching process in logic programming, 3. the ERM2C methodology, 4. the results achieved from applying the methodology, and 5. the conclusion.

2 THE PROBLEM WITH THE LEARNING/TEACHING PROCESS IN LOGIC PROGRAMMING

After more than 22 years of programming experience and 2 additional years of teaching experience in higher education institutions, I had my first experience as a professor of Logic Programming at a university in 2000.

At the end of the first semester I was concerned with the results. More than 60% of my students had failed the course because they were not able to build algorithms. I concluded that, despite having experience in logic

programming, I was not able to teach it using the methodology found in the list of textbooks recommended for the subject. Therefore, I decided that another strategy had to be found for teaching the subject.

During the second semester, after including other teaching strategies, the failure rate was still high at 50%. This led me to question my ability to teach logic programming but also to wonder if the problem was just me.

Informal surveys of logic programming and other similar courses performed during the last eight semesters in 4 local universities ascertained that more than 60% of students failed these courses each semester (Table 1). It became clear that problems teaching the subject were not exclusive to me. Next, I questioned if the problem was local.

Table 1 – Logic Programming Failure Rate at Local Universities

Institution	Semester							
	1st	2nd	3rd	4th	5th	6th	7th	8th
A	57,14%	46,15%	33,33%	46,15%	33,33%	46,15%	33,33%	18,18%
B	35,33%	57,14%	75,00%	33,33%	75,00%	33,33%	57,14%	75,00%
C	75,00%	70,58%	57,14%	46,15%	88,88%	46,15%	75,00%	75,00%
D	88,88%	57,14%	33,33%	75,00%	87,50%	75,00%	57,14%	33,33%

The answer is that the subject of logic programming has one of the highest rates of failure in universities [1][2][3][4][5][6][7][8], which makes teaching it one of the seven greatest challenges in computer science education of this century [9]. Therefore, it is imperative to determine the reasons for this problem and subsequently identify strategies that can improve the learning/teaching process.

Private interviews with over 60 students from some of the universities surveyed show that the vast majority of students have trouble building algorithms for even less complex problems (Figure 1).

Figure 1 – Learning difficulties in Logic Programming

Difficulties in problem solving are related to [2][3]: (i) the level of detail the solution must have; (ii) associating procedures established in natural language to procedures in programming language.

The second problem can be addressed by applying rules that restrict an algorithm's set of possible actions to an equal set of standard commands that are used in most programming languages [10].

The problem of the level of detail is related to skills acquired from studying and from practical experience [11]. However, it is not easy to acquire experience when building solutions is a problem.

One possible answer can be found in strategies adopted by professors who have lower failure rates. With this in mind, professors from a number of universities were interviewed in an attempt to identify the methodologies they use to teach the subject.

The results showed that most professors followed the procedure found in most of the textbooks recommended for the subject (Figure. 2): building algorithms to solve problems. Additionally, some professors taught their students to evaluate and/or correct, and/or update, and/or optimize solutions before teaching them how to build it.

Figure. 2 – Logic Programming Teaching Strategies

The issue, therefore, is: there is no guarantee that students will learn and that failure rates will decrease when professors teach the method of building algorithms as the way to solve posed problems, which is difficult for students to accomplish.

There is no easy answer to this issue. However, if we consider that studying and practical experience produce the necessary skills, viable alternatives can be found in methods such as the ERM2C, which encompasses all of the practices identified in our research: (i) evaluate existing solutions; (ii) maintain existing solutions by either correcting the problem or updating the solution; (iii) optimize existing solutions; and (iv) create new solutions.

3 THE ERM2C METHODOLOGY

Professors and researchers use three approaches to improve the learning/teaching process in logic programming: Tools, Strategies, and Tools and Strategies together. The ERM2C methodology is one of these strategies.

This methodology considers the main objective of logic programming, which is for students to be able to develop an algorithm, or a finite sequence of actions to solve an established problem by the end of an academic period [12][13][14][15][16][18][19][20][21].

According to Chantler [17], a good program using clear algorithms has the following characteristics: it does what is expected (precision) continuously (reliability); it manipulates all data precisely and correctly (power) without any unnecessary processing (efficiency); and is easily altered (maintainability) [17][19].

Because most students have little or no experience, it is sometimes nearly impossible for them to make build algorithms that are precise, reliable, powerful, efficient, and maintainable enough to produce good programs.

Therefore, before developing algorithms students must be able to easily read and understand them, whether theirs or another's, or whether the problem is defined or not. Having good examples is a step toward building products that are similar or better.

The ERM2C methodology was based on this premise. It is made up of 5 stages (Understand, Review, Improve, Complement, and Build) and each of the stages is divided into 3 levels (basic, intermediate, advanced). The stages were grouped into 2 phases: algorithm knowledge and algorithm building, both partial and total.

Students advance levels when an established minimum number of exercises are correctly concluded. Once students finish the advanced level activities, they are qualified for the next stage.

The algorithm knowledge phase begins with the Understand stage, where students learn to read and understand algorithms, independent of their author. These skills are indispensable in the Review stage where students identify the combination of actions that produce unexpected final results within an algorithm.

In the Improve stage students propose modifications to algorithms with the objective of improving its performance.

Once the algorithm knowledge phase is concluded, students are qualified for algorithm building. Here students acquire skills for the Complement stage that deals with completed logic programming and seek to develop or adapt solutions to meet newly identified needs.

The Build stage in algorithm building is where students create their own solutions to stipulated problems, or in other words, they build their own algorithms.

Before the algorithm knowledge and building phases can be developed, however it is necessary to have a preliminary phase called Introduction, where the basic concepts of algorithm logic and graphic representation methods are presented.

Because the ERM2C is "a methodology that preserves the individual characteristics of the students, since it is individual development as opposed to group development, that allows level advancement within each stage"[3] (each class with, maxim, 30 students), the professor must identify each student's level of experience at the end of the introduction phase.

3.1 Identifying Student Experience

Identifying student experience is done through an evaluation of five questions, each referring to one stage of the ERM2C. All questions are at intermediate level.

The objective evaluation is to determine the kind of experience students have with logic programming development in order to establish their starting level in the course. This keeps students that have more experience motivated to develop their projects.

At Jeriz's suggestion [22], the stage, the level within each stage and suggested requirement for releasing students from the class are presented in Table 2.

Table 2 –Student Stage and Initial Level

Stage	Level	Result of Evaluation
Understand	Basic	Did not conclude any of the 5 questions.
	Intermediate	Did not conclude the last 4 questions and finished less than half of the first.
	Advanced	Did not conclude the last 4 questions and finished more than half of the first.
Review	Basic	Did not conclude the last 4 questions, but completed the first.
	Intermediate	Did not conclude the last 3 questions and finished less than half of the second.
	Advanced	Did not conclude the last 3 questions e finished more than half of the second.
Improve	Basic	Did not conclude the last 3 questions, but finished the first 2.
	Intermediate	Did not conclude the last 2 questions and finished less than half of the third.
	Advanced	Did not conclude the last 2 questions and finished more than half of the third.
Complementary	Basic	Did not conclude the last 2 questions, but finished the first 3.
	Intermediate	Did not conclude the last 2 questions and finished less than half of the fourth.
	Advanced	Did not conclude the last 2 questions and finished more than half of the fourth.
Build	Basic	Did not conclude the last question, but finished the first four.
	Intermediate	Concluded the first four and less than half of the fifth.
	Advanced	Concluded the first four and more than half of the fifth.
Released from subject		Correctly concluded all 5 questions.

3.2 Phase: algorithm knowledge

The **knowledge** phase encompasses the first three stages of the methodology (Understand, Review, and Improve) due to the belief that before students can start developing an algorithm they must be able to:

1. Interpret algorithms, independent of the author (Understand).
2. Recommend algorithm actions that can alter the final expected result. (Review).
3. Propose alterations to correct wrong actions or include new actions to improve the performance of the algorithm or to alter its main objective. (Improve).

In the **Understand** stage, students read completed algorithms that have been validated by other people with more experience, or in other words, they have no logic mistakes. Students answer a series of questions related to the results expected at the end of processing the problem. There are typically at least 10 exercises for each problem level (simple, medium, and complex).

The goal of the **Review** stage is for students capable of understanding an algorithm to identify the combination of actions that caused unexpected final results (**corrective maintenance**). Students will usually need problem descriptions, incorrect solutions, and the problem that was detected in order to identify the problematic areas. At least 10 exercises in each level are necessary to successfully prepare students.

One of the most complicated stages is **Improve**, which is divided into two distinct parts: identifying and removing unnecessary steps to achieving expected results in algorithms (**power**), and identifying a combination of actions that, despite helping to achieve the expected results, reduce algorithm performance, and proposing a combination of actions to replace them (**efficiency**). Each part of the stage must be worked on separately and a minimum of 10 exercises for each level of complexity is required.

3.3 Phase: algorithm building

The building phase corresponds to 2C in the ERM2C methodology. It is similar to what is found in textbooks, but different in how solution-building process as a whole is conducted. This phase is divided into the **Complement** and **Build** stages.

In the **Complement** stage, students should already have the ability to understand, review, and improve incomplete algorithms that still need developing or adapting due to the alterations in initial objectives. Students now propose a combination of actions that are needed to achieve the new expected results. In this stage, **ongoing** and/or **adaptive maintenance** can be used, initially with only one logic point, and later in various places in the algorithm. In both cases, a minimum number of 10 exercises for each level of complexity are required.

The **Build** stage is the same one used in traditional methodology where students develop a logical solution to a posed problem. The number of exercises must not be less than 10 per level of complexity.

4 ERM2C METHODOLOGY: RESULTS

The ERM2C Methodology was taught during four semesters to associate degree students enrolled in logic programming classes in two higher education institutions. It was offered during alternative time slots and totaled 48 class hours.

The students who participated volunteered for the experiment and were divided into 2 groups (each one with 11 students/semester): students taking the subject for the first time and students who had already taken the subject and failed. Each of these groups worked at different times, and was not allowed to switch times.

Table 3 – Students Approved in Logic Programming – ERM2C Methodology

Institution	Semester							
	New	Returning	New	Returning	New	Returning	New	Returning
A	7	9	8	10	9	10	10	11
B	8	10	9	10	10	11	11	11

Although inconclusive, the final results (Table 3) can be considered satisfactory. The number of new students who failed was on average 2 students, or 18% lower than the 37% shown in the research. Among returning students the failure rate was less than 1 student on average or approximately 8% lower than the 23% shown in the research.

The methodology was not continued after the conclusion of the fourth and last semester due to trouble finding new volunteers and the author's departure from teaching.

However, the methodology was again used in a private, regular course of Logic Programming during the second semester of 2009. It was taught in 48 class hours, over a period of 3 months that consisted of 4 monthly meetings that lasted 4 hours each.

The group had 4 new students and 8 returning students, all of which were enrolled at higher education institutions in ITC. In this group only 25% failed, including 2 students who dropped out of the ITC program.

If the students who dropped out are not considered, the approval rate was 100% for new students and over 83% for returning students.

5 CONCLUSION AND FUTURE ENDEAVORS

Most ITC university students consider logic programming as being the most difficult subject, which is proven by the high rate of failure at the end of each semester. Therefore, teaching it is considered one of the seven greatest challenges in computer science education in this century.

Some professors and researchers have been working with three approaches to overcome this challenge and improve this teaching/learning process. These are: tools, strategies, and tools and strategies.

As a strategy, the ERM2C Methodology initially prepares logic programming students to increase their knowledge of algorithms, and later enables them to build solutions to problems. These are called the **knowledge** and the **building** phases.

During the **knowledge phase**, students acquire the skills necessary to understand (the ability to read), review (the ability to read and identify incorrect actions that must be removed or replaced) and improve (the ability to read, identify problematic actions, and to propose improvements) algorithms.

During the **building stage**, students acquire the skills necessary to complement (identify and propose changes that meet the needs of the alterations in problem definitions) and build (propose a solution for the resolution of a problem) algorithms.

The methodology was applied to small group of volunteer associate degree students during 4 semesters, and for 1 semester in a regular private course. The rate of student failure among the participants was lower than the 60% rate identified in the informal research.

This result, albeit satisfactory, is not yet conclusive. The author, however, believes that the ERM2C methodology can be used in logic programming courses offered in undergraduate or technical programs in the ITC area.

Since it is new, future work is needed to consolidate the methodology:

- Implement the methodology for logic programming in other private courses over the next few years
- Include the methodology in logic programming courses for associate's degree volunteer students
- Specify a system for measuring student success rates (approved or failed) at the end of each school semester, considering the last levels and stages concluded by students
- The conclusion and publication of 2 books about the methodology, one that presents the theory and the other that presents practical exercises.

6 INFORMATION AND QUESTIONS

You may contact at rluizcampos@yahoo.com.br

REFERENCES

- [1] Raabe, A. L. A. e outros. Adquirindo experiência na construção de ferramentas de apoio a aprendizagem de algoritmos. Workshop de Ambientes de Apoio a Aprendizagem de Algoritmos e Programação. Simpósio Brasileiro de Informática na Educação, 2007. Anais dos XVIII Simpósio Brasileiro de Informática na Educação (2007).
- [2] Campos, R. L. B. L. Lógica de Programação: Há como melhorar o aprendizado fugindo dos padrões estabelecidos nos livros didáticos e adotados pela maioria dos docentes?. XVII Congresso Iberoamericano de Educación Superior em Computación (CLEI-2009-CIESC), 22-25/set/2009a, Brasil, RS, Pelotas (2009).
- [3] Campos, R. L. B. L. ERM2C: Uma proposta de metodologia para melhoria do ensino-aprendizado de lógica de programação. XI Congreso Chileno de Educación Superior en Computación (CCESC), 2009b. Santiago, Chile, Jornadas Chilenas de Computación 2009. Santiago, Chile, 2009b, V. Único (2009).
- [4] Marczak, S. dos S; Giraffa, L. M. M.. Ambientes Inteligentes para Suporte ao Ensino de Programação. Technical Reports Series, Number 28, August, 2003, Faculdade de Informática – PUCRS (2003).
- [5] Dos Santos, R. P.; Costa, H. A. X.. “Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática”. Infocomp, Journal of Computer Science., Volume 5, Number 1, March 2006. Universidade Federal de Lavras – UFLA, Lavras, MG, Brasil (2006).
- [6] Fackembach, G. A. M.; De Araujo, F. V. “Aprendizagem de algoritmos: dificuldades na resolução de problemas”. Universidade Luterana do Brasil – ULBRA, Santa Maria, RS, Brasil. Faculdade Dom Alberto, Santa Cruz do Sul, RS, Brasil.
- [7] Pereira Júnior e outros. “Ensino de algoritmos e programação: uma experiência no nível médio”. XXV Congresso da Sociedade Brasileira de Computação, Unisinos, São Leopoldo, RS, Brasil, 22 a 29 de Julho (2005).
- [8] Rapkiewicz, C. E. e outros. “Novas Tecnologias na Educação”. CINTED-UFRGS, V. 4 Nº 2, Dezembro (2006).
- [9] Wesley, H.; Gondim, A. P.; Ambrosio, A. P. Esboços de fluxogramas no ensino de algoritmos. Workshop sobre Educação em Computação – WEI. Anais do XXVIII Congresso da Sociedade Brasileira de Computação. Belém do Pará, PA, Brasil, 12 a 18 de julho de 2008.
- [10] Pyott, S., Sanders, I., “ALEX: An aid to teaching algorithms”. SIGCSE, Vol. 23, Nº 3, Bulletin, Set/1991, pag. 36-56.
- [11] Villas, M. V. e outros. “Programação: conceitos, técnicas e linguagens”. Rio de Janeiro: Campus (1987).
- [12] Forbellone, A. L. V., Eberspächer, H. Z. Lógica de programação: a construção de algoritmos e estrutura de dados. 3ª edição. São Paulo: Prantice-Hall (2005).
- [13] Farrer, H. e outros. Programação estruturada de computadores: algoritmos e estrutura de dados. Rio de Janeiro: Editora Guanabara (1985).
- [14] Villas, M. V. e outros. Programação: conceitos, técnicas e linguagens. Rio de Janeiro: Campus (1987).
- [15] Campos, R. L. B. L. Lógica de programação (apostila). Brasília: (2001-2003-2004).
- [16] McConnell, J. J. Analysis of Algorithms: an active learning approach. USA: Ed Jones and Bartlett Publishers (2001).
- [17] Chantler, A. Técnicas e práticas de programação. Rio de Janeiro: Campus (1984).
- [18] Mazano, J. A. N. G.; Oliveira, J. F. de. Algoritmos: lógica para desenvolvimento de programação. 21ª. São Paulo: Érica (2007).
- [19] Campos, R. L. B. L. Lógica de programação – DHF. Módulo 4: curso de capacitação em programação. Brasília: SERPRO (1993).
- [20] Farrer, H. Algoritmos estruturados: programação estruturada de computadores. Rio de Janeiro: Guanabara (2000).
- [21] Guimarães, A. de M. Lages, N. A. de C. Algoritmos e estruturas de dados. 8ª edição. Rio de Janeiro: LTC (1989).
- [22] Jeroz, R. “Sugestão para a avaliação do alunos no início do semestre”. Universidade Católica do Norte do Chile. Conversa informal durante as Jornadas Chilenas de Computación de 2009.