Didactic Framework for Teaching C Programming Language: A Proposal Based on Cooperative and Competitive Learning Techniques

Wollace de Souza Picanço
DEPEG-UFAM
Federal University of Amazonas
Manaus, Brazil
wollaceps@gmail.com

Juliana M.V. Martínez de Lucena
CMC-IFAM
Federal Institute of Amazonas
Manaus, Brazil
juliana.lucena@ifam.edu.br

Antônio Fonseca de Lira
CMC-IFAM
Federal Institute of Amazonas
Manaus, Brazil
antonio.lira@ifam.edu.br

Vicente Ferreira de Lucena Jr.
PPGEE-PPGI-UFAM
Federal University of Amazonas
Manaus, Brazil
vicente@ufam.edu.br

***Abstract*— This Research to Practice Full Paper presents a didactic framework for teaching C programming language. The difficulty of understanding abstraction presented by engineering students in computational problems solving is one of the main reasons for numerous pitfalls in the early college semesters. When teaching abstract contents, such as computer programming, it is relevant to include as many practical exercises and laboratories experiments as possible, to facilitate a better understanding. These practical approaches allow the students to achieve, by simulations or by real experiences, good notions of the basic concepts of the subject being studied. To motivate engineering students to be acquainted with formal definitions and high-level abstractions, a Didactic Framework focused on the teaching of the C programming language for embedded systems was developed and tested. Moreover, the students were supposed to develop a software project in teams that work together to solve basic tasks, and that compete at the end of the semester by solving a common programming challenge. This work is divided into two main parts: one focusing on the creation of practical strategies that make easier understanding complex definitions; and another concerning to the motivational strategies through cooperation and competition. The final goal was to increase the students understanding and satisfaction as well as reduce the dropout rates.**

Keywords*— *Learning with Robots; C Programming Language Learning; Collaborative Learning; Competitive Learning.

I. INTRODUCTION

According to a recent survey conducted at the UFAM in Manaus, Brazil, the difficulty in understanding programming concepts, like abstraction, decomposition, or forming perspectives, among engineering students in the first college semesters has caused numerous pitfalls and led to an increase of dropout rates. This perception confirmed the findings of many authors [1, 2, 3, 4]. One common reason for such problems is that traditional teaching approaches are based on methodologies in which the students are passive along the learning process, what makes the deep understanding of abstract concepts difficult and demotivating [1, 4, 5]. In fact, that is not a local problem, many authors have documented a big dropout rate in computer science and engineering courses related to bad comprehension of basic topics [6, 7, 8].

SAMSUNG, CETELI, CNPq, CAPES, and FAPEAM supported this work.

Many education theories may be adopted in this quest for improving the learning process for computer related themes. For example, Constructivism considers that students must be active in the learning process, building their knowledge from experiences in a day-to-day basis and contact with the world and with real objects [9, 10]. There are many streams of Constructivism, but almost all of them come from the work of Jean Piaget, who focused on how humans make meaning about the interaction between their experiences and their ideas [11]. The concept of Constructivism has influenced several disciplines, including psychology, sociology, education, and more recently, computer programming. In the very beginning, Constructivism examined the interaction between human experiences and their reflexes or behavioral patterns, what was called knowledge schemes. Constructivism does not refer to a specific pedagogy, although it is often confused with constructionism, another educational theory inspired by constructivist and experiential learning ideas of Piaget.

Maria Montessori elaborated a theory based on stimulation through the manipulation of objects and affirmed that education material should be created based on the learning goals and offered to the learners that would explore them, performing various tasks in their path [12, 13]. The original method is a child-centered educational approach based on scientific observations of children from birth to adulthood [14]. The method views the child as one who is naturally eager for knowledge and capable of initiating learning in a supportive, thoughtfully prepared learning environment.

Pragmatism favors the resolution of problems reasonably and logically based on dealing with specific situations instead of on abstract ideas and theories [15]. Pragmatism considers thought as an instrument or tool for prediction, problem-solving and action, and rejects the idea that the function of thought is to describe, represent, or mirror reality. Pragmatists contend that most philosophical topics—such as the nature of knowledge, language, concepts, meaning, belief, and science—are all best viewed regarding their practical uses and successes [16]. The philosophy of Pragmatism emphasizes the practical application of ideas by testing them in human experiences. Pragmatism focuses on a changing universe rather than an unchanging one as the Idealists, Realists, and Thomists had claimed.

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

In fields like computer programming, which demands more abstract thinking, for example, it is relevant to include practical teaching approaches in the sense of the three theories mentioned above [17]. We need to offer a learning environment to satisfy the eager of knowledge from our junior students just like Montessori claims (although we are not working with children). The learning objects must reflect the pragmatism of real-life projects to keep the students focused on the learning goals, and it is necessary to fix the students' knowledge from their findings based on the contact with real objects. These approaches can play a key role in a better understanding because they allow the materialization of concepts and help to simulate abstract definitions through practical exercises. Object-oriented programming languages, for example, involve formalisms and other complicated conceptual abstractions that may be very difficult to understand without an active approach, access to proper learning material, and by doing practical experiments and real-world exercises [18]. In fact, there are many other obstacles in teaching programming languages such as the difficulty of understanding the language syntax, the complex structures of algorithms, and other fundamental concepts. All of them must be considered when planning an appropriated teaching-learning methodology [19].

Similar problems occur when teaching programming for embedded systems. This kind of system is an integral part of many devices in modern life, as it specifically targets the control of widespread products like mobile phones, digital TVs, games, robotic objects, and many other devices. That is a very important area in new electrical and computer-engineering curricula all over the world. Those curricula cover a bunch of multi-disciplinary fields and frequently start teaching programming by using the C Programming Language.

Having this context in mind and aiming to, in one hand minimize the described problems, and on the other hand, maximize the motivation of new students entering our engineering courses, a Didactic Framework was developed at UFAM and will be presented here. This Framework is already in use in first-year classes for Electrical Engineering, Computer Engineering, and Computer Science students. The courses are mostly four credits, what means 60 hours per semester and the students have no previous programming courses at their program. The specific learning goals of this course is to domain the C programming language, which is normally the first approach to programming computers at our university. It is also expected that the use of this tool helps to reduce the dropout rate. At the end of the semester, the students are supposed to develop a software system in teams and compete by solving a common programming project. The paper is divided into two parts, one presenting the Framework designed to help lecturers and students in the teaching-learning process, and another describing the adopted motivational methodology.

In fact, the proposed Framework tries to minimize the difficulty of understanding abstraction concepts (feeling, intuition, perception, and thought) presented by new students. It was necessary to develop a user-friendly tool containing the core subjects of a specific programming language, its commands, libraries of reusable functions, and other relevant details. In order to reduce the abstraction problems, a commercial mobile robot was used as a learning platform. The idea of using a robot raised from the premise that these devices are designed and engineered to help humans to perform certain tasks. It should be easier to recognize such tasks and to program the robot to solve them. In fact, when interacting with the real object using its components (controllers, sensors, actuators, manipulators, gears, etc.), the students can establish a relationship between the real world and their intellectual perception, i.e., improvement in their cognition. Also, there are many positive reports on the use of robots as learning tools.

The second part proposes a way to motivate the students through the techniques of Collaborative Learning and Competitive Learning. These approaches are well-known methodologies in education, which arise from the need to insert interactive procedures for students in conjunction with the educator to establish the understanding and interpretation of information for specific issues [19, 20]. Some authors have pointed out negative aspects of these techniques [21, 22], while others have pointed out that they motivate the students, allow the exchange of experiences, arise better understanding, and develop diverse teamwork competencies in an active approach [23, 24]. In this phase, a certain mission is proposed for the students; they are divided into groups that should complete the mission, and that compete towards the best solution. Their step-by-step work is graded according to the obtained results and the quality of the delivered software. At the end, just like in any other real competition, the one that better meets the needs and expectations of the client (lecturer), in respect to their specific mission, is considered the winner.

In resume, the main objective of this work is to present a didactic framework capable of minimizing the lack of abstraction and motivation of students in the classroom, more specifically, in the disciplines of software programming language for embedded systems. This paper is structured in the following manner: Section II – contains the related work and specifies its motivation; Section III – show details of the proposal; Section IV – evaluates the obtained results; and Section V – presents some final remarks and future work.

II. Related Work and Motivation

Many educators are researching the proper way of using technological tools to minimize difficulties in the teaching-learning process [25, 26, 27]. In many underdeveloped countries, the education system is experiencing a major crisis. The OCDE (Organization for Economic Cooperation and Development) maintains a ranking of education in 36 countries, in which the authors' country currently bitter one bad position. Among the main criteria evaluated by the OCDE are the performance of students in PISA (score and position in the world ranking), the mean number of years that students spent in school, and the percentage of the population enrolled in higher education [28]. In general, this tough situation of the fundamental school reflects on the universities, mainly in the first year of higher education.

A well-designed framework is one that focuses on the faced problems and facilitates the usage of the existing tools. When such a tool is incorporated into undergraduate curricula, it will certainly help lecturers to prepare better classes training the

students for problem-solving projects [29]. It is also demonstrated that well-planned learning competitions among students provide an elevated level of understanding [30]. Reported results showed that more than 95% of such learning teams had applied their basic acquired knowledge properly in embedded systems and robotics, 60% of teams have applied their knowledge to develop a solution for a problem, while more than 30% of the teams could critically analyze the problem effectively. The preliminary results show that such methodologies are attractive and can, in fact, motivate the students.

Many researchers reported positive impacts of using robots or another device to keep the students' attention and to facilitate the understanding of complex concepts [31, 32, 33, 34, 35]. Student competitions also provide an elevated level of understanding of the learning subjects and improved the students' satisfaction [2, 17, 21]. It also helped students and lectures to focus on the real learning objectives. Nevertheless, robots are not the only one existing artifact to improve learning of computer-related subjects [36, 37, 38].

Huggard and colleagues reported that one of the most relevant changes in the curricula of engineering courses had been the emphasis on the development of personal skills [39]. These skills include the ability to learn through collaboration and critical reflection of the individual behavior [40, 41]. This research reports on how cooperative learning was integrated into practical modules in engineering courses for first-year students. The authors explored the practical experiences of students to encourage them to increase their significant reflection. Other studies detail phenomena that influence the attitudes of engineering students about cooperative and reflective learning [42, 43].

Based on this assumption and motivated by these studies, this work had two main goals. The first goal was to develop a framework to be used in the practical part of a programming language course and the second goal was to create scenarios that implement the motivational methodology, where the framework could be used as a practical tool common to all participants.

III. Description of the Developed Framework

The proposed framework was designed to minimize the difficulty of understanding abstraction and other computer science related topics, deficiencies normally identified in students entering University. The framework should have a user-friendly interface containing helpful features to lecturers and students, and its technical content should cover core subjects of a specific programming language, its commands, libraries of reusable functions, and other relevant aspects. It should also be tied to a practical approach designed to help the learning of concepts and to increase the overall learning curve.

There are diverse commercial tools with those characteristics that could be used partially in this work. Nevertheless, it was desired to develop an evolvable framework, i.e., one that the developers could modify based on new lecturers' demands. Most commercial tools do not accept that possibility, and the authors decided to develop one by themselves. On the other hand, the construction of the practical device to be used along the course needed to be robust and precise. It should not be a problem by itself, and the choice of the authors was a didactic commercial moving robot.

The teaching methodology aims to keep the students focused on the learning objectives and their assignments. Inspired by some good reports in the literature, a mixed approach containing characteristics of Collaborative Learning techniques [38, 44, 45, 46] and Competitive Learning techniques [37, 40, 41, 47, 48] was elaborated. Those techniques motivate the students, allow the exchange of experiences, promote better understanding, and help to develop competencies more quickly [23, 24]. This proposal was applied to a group of students, and their behavior along the classes, satisfaction towards the framework and about the methodology were observed, measured, and evaluated. In the next sections, details of the developed work will be given, as well as, comments on the evaluation results.

In fact, by achieving the overall objective, it was expected to develop four specific capabilities of the involved students. They are:

- Cognition – to develop in the students their capacity for feeling, intuition, and perception of opportunity.
- Competence – awakening abilities in students, to put into practice the theories and mental concepts that were acquired during the course.
- Competitiveness – to improve in the students the capacity to formulate and implement strategies, under pressure, thus allowing a privileged status over the other teams.
- Leadership – to develop the students' skills of a leader involving charisma, patience, respect, discipline and, especially, the ability of positively influence subordinates.

A. Proposed Framework

Fig 1. shows the main elements of the proposed framework. The first is the Communication Device; it is the mediator between the students and the core of the Framework (through a User Interface). The Core Module contains a library of reusable components focused on the C Programming Language and the Target Embedded System. The third element is the Embedded Systems itself which may be any device with known interfaces, and that runs C programs. In this first approach, it was a moving robot. The framework runs at the top of the operating system aiming at manipulating actuators and sensors on the robot, offering a well-defined user interface, and giving the students appropriated material about the learning object.

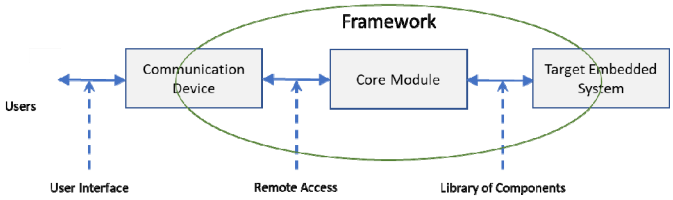

Fig. 1. Component Diagram of the Framework proposed.

In this first approach, the framework covers all aspects of the C Programming Language. Examples dealing with fundamentals of the language structure (selection, decision, loops, etc.) were inserted on the framework. The same was done for other important themes like program control directives, common C functions, how to work with arrays, with pointers, and with Strings.

The inserted didactic material covers the subject needed for first-year students. By solving a typical example implies not only in printing the result on the computer screen but also on some action on our robot.

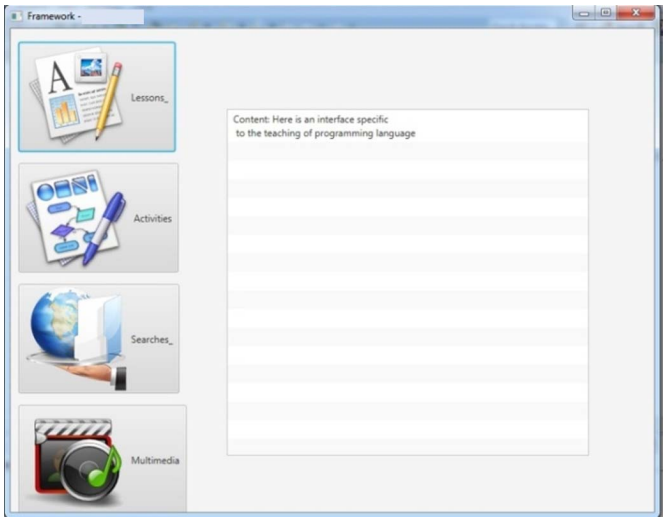

Fig. 2. Initial Interface of the Framework.

Fig. 2 shows the initial screen of the didactic framework; it makes the interaction between the User and the Embedded Systems. Its menu displays and allows the selection of learning tools for the teaching process, composed of the following items:

- Lessons – with theoretical specific content to explore and with the option to insert new material for the teaching of programming languages. A first attempt was made with the C programing language. Fig. 3 shows an exemplary screen of this area.
- Activities – this area contains practical exercises focused on the desired learning topic. Lecturers are free to insert guided procedures, and the students are supposed to conclude their assignments properly. The students may make use of pre-defined commands and other options to control the robot or may program it directly in C. Fig. 4 contains one example for the robot commands. The Framework has the options Commands, Execution, and Simulation. The components of this option are input and output commands, decision structures, and repeat structure. It also contains initial default Variables that may be modified during the experiment. The Operators palette offers arithmetic, logical and relational options on the C Language, and in the Movements palette some functions tied directly to the robot behavior. After the creation of a project, the students may simulate or execute a robot command. For the simulation, the area where the robot may move was delimited through the showed Cartesian plane, that is, in the x-y axis and has a maximum of 480 units wide by 360 in height. The black dot on the grid shows the robot. The real execution of the project runs on the real robot that moves on the laboratory floor.
- Searches – an area designed to direct research commands, programs, and other resources on the internet. Predefined links will help students to find programming alternatives. Fig. 5 contains a screenshot of this part of the Framework.
- Multimedia – this area contains text, audio and video resources related to the programming language. It also offers one Help menu with access to users' manual (Fig. 6).

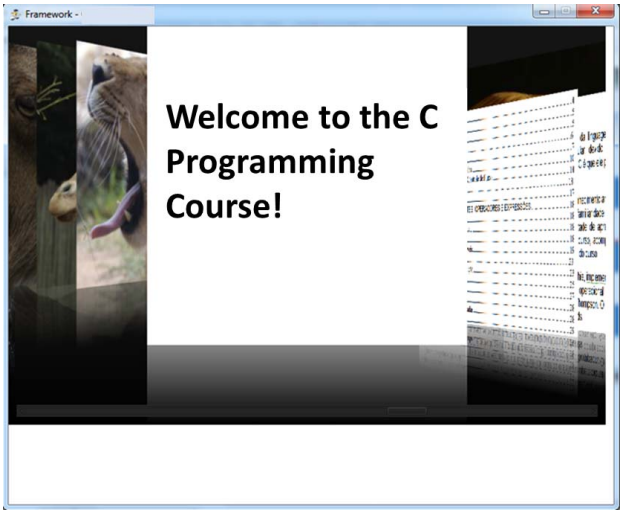

Fig. 3. Component Diagram of the Framework proposed.

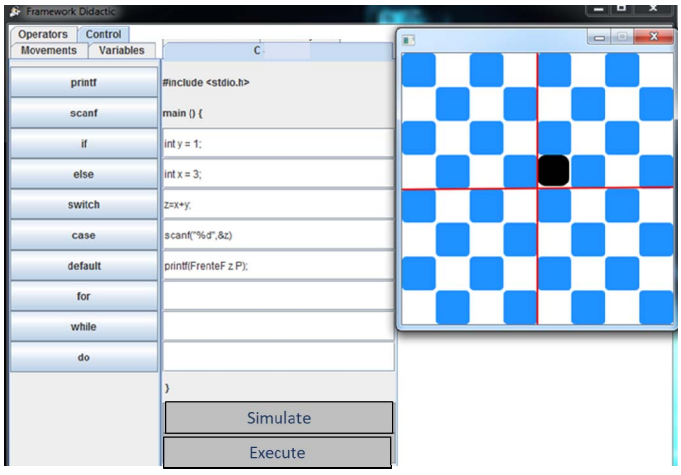

Fig. 4. Library of Components of the Didactic Robot.

A communication interface was also implemented between the core module and the embedded device. This interface makes use of a transport primitive (Socket) of type SOCK_STREAM (data travels in the form of stream characters) to implement the interface commands in the framework. The Ethernet standard 802.11 is used to perform the interconnection and to create one Access Point (AP) between the computer and the robot.

Fig. 5. Search Part of the Didactic Framework.

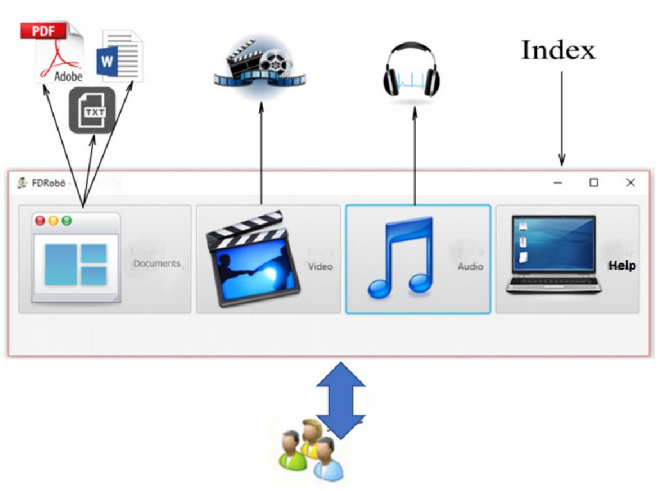

Fig. 6. Multimedia Part of the Didactic Framework.

Students should not worry about the programming of the robot itself. They should focus only on the aspects of the C programming language. A set of guiding libraries, which reuse software components of the robot (controllers, sensors, actuators, handlers, etc.) were integrated into the framework. They are responsible for executing commands for moving the robot to different directions (right, left, front, and back). Fig. 7 contains one robot command example. The proper choice of commands and parameters from the library of the Didactic Framework is responsible for managing the basic movement of the robot. Correct usage of the C programming language will result in the proper robot command execution. In fact, the in C written commands will access the application-programming interface (API2) of the robot and will trigger the necessary component. API2 also gives full access to robot's sensors and actuators. Using this platform is easier to learn the C directives because you can confirm that a correct C program resulted in a desired robot's behavior. For example, if the students need the robot to turn left or right, they will make use of directives of the framework to move the robot. After this configuration, the framework interprets the in C language written programs, formats the requested command in accordance with the robot directives, and sends the command to the robot that should behave properly. Other pre-defined functions are responsible for setting coordinates, establishing moving speed, performing detection of objects in the route, etc.

The mobile robot used in this project was the Robotino® from Festo Didactic (Fig 8). The robot is composed of three omnidirectional wheels, arranged at angles of 120°, which are individually controllable. The Robotino® has many sensors like light sensors (infrared), impact sensor (also called Bumper), color sensor, inductive sensor, and one image sensor (a camera with VGA resolution). Additionally, it contains optical encoders wheels, power measurement for the entire system and the various engines, as well as a monitor of battery charge status. It can also be equipped with additional peripherals, like a laser scanner, a gyroscope and a tracking system in interior spaces. The robot has several interfaces like USB, Ethernet, eight digital inputs and eight analog ones, eight digital outputs, additional engine power for driving high loads, and entry of additional encoders [49]. The most important characteristic is that it is reliable and robust enough to support the curiosity of students.

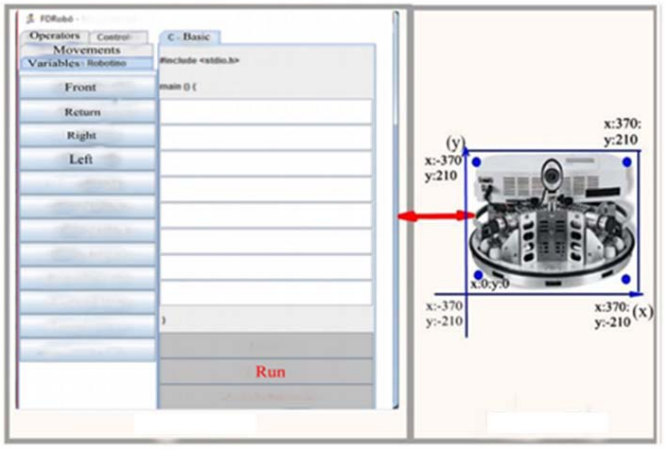

Fig. 7. Robot command example.

Fig. 8. Robotino Festo Didactic [www.festo-didactic.com].

B. Motivation - Learning Techniques

The proposed methodology has two main events. In the first part, the students are supposed to learn in groups. The lecturer sets up the learning goals and asks students to develop the desired theme by themselves. Those activities must be executed in groups and teammates are supposed to work together helping each other. During the whole semester, the lecturer is always available to help students. The second part consists of a competition among teams. In one pre-scheduled date, the teams meet in the laboratory, and a set of assignments are distributed. They should solve the problems and points are given based on the correctness, style, and time of the resulting solutions. In the end, one team is declared a winner.

- Cooperative Learning

The adopted cooperative learning approach was based on the assumptions presented by Huggard [39], but the didactic framework was adopted here, as the only tool to support the teaching-learning process. Student teams were formed at the beginning of the course. One leader per team was chosen, and they organized, together with the lecturer, the desired activities for the rest of semester. These leaders selected the rest of the members of their teams through a stochastic rotation system. Some basic rules for the formation of the teams were:

- Each student should take part of only one team;

- Each group must have a name chosen by consensus among all members;

- The leader is responsible for presenting the results of the team, and for the coordination of all activities; and

- The teams should use the didactic framework to solve the proposed problems.

The lecturer defines the meetings frequency, and the members of the team must propose a set of activities that cover all the desired learning topics. They should plan their learning schedule, as well as propose guidance meetings to resolve possible conflicts. The lecturer must define the schedule and the location of the class meetings and offer alternatives in the event of unexpected changes.

In general, any other aspect that can be considered important for the development of activities must also be specified. The objective of establishing rules is to allow students to decide by themselves team procedures on how to cooperate. Finally, the lecturer makes sure that all team members are uniformly involved in the learning activities.

After the definition of the organizational structure, a set of theoretical problems is proposed to each team. These problems were designed to help the students to achieve the fundamental concepts of the programming language being learned. After the teams have solved the problems, the second phase of the learning process starts. This phase is based on a competition among the participating teams. Fig. 9 summarizes the cooperative learning process.

- Competitive Learning

The teams meet in the laboratory and prepare themselves for the competition. The correct solution of diverse common tasks results in the final score of the team. A better score will be given to the team that presents a better result. During these activities, the lecturer can select one of the team members to explain the proposed solution to the rest of the participants. This student will be responsible for the team's obtained grade. Fig. 10 represents the Competitive Learning approach.

A set of activities, theoretical as well as practical, are distributed to the competitors. They should solve the problems together in their teams and present the proposed solutions to all participants. One particular set of activities involves the control of the mobile robot by using the didactic framework.

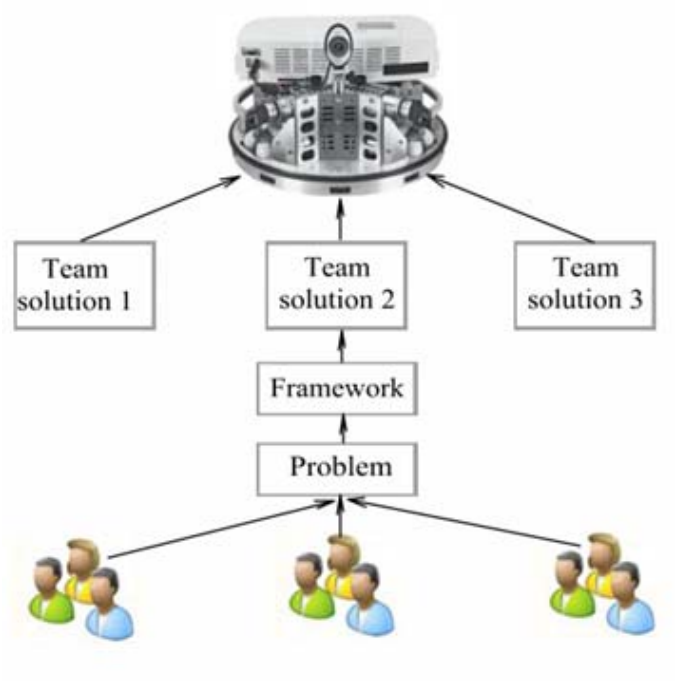

Fig. 9. Cooperative Learning Process.

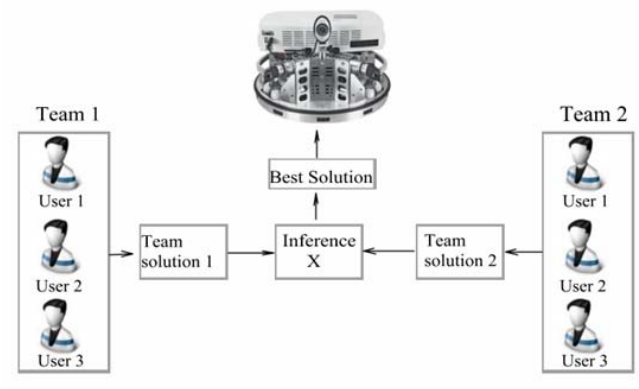

Fig. 10. Competitive Learning Approach.

The most exciting competition event is the one with the mobile robot. In our last semester, the teams were supposed to develop an algorithm to complete a racetrack. The winner was the team with the most efficient algorithm, the one that completed the track in shortest time.

The final score of each team is generated by adding the average score of each activity and the score obtained at the robot race. At the discretion of the lecturer, there will be one or more winners on diverse categories of solved problems. The lecturer can decide if only the best team get extra points or if all the teams that presented outstanding results deserve points as a prize. Moreover, at the end of the activities, they are supposed to celebrate together for getting the job done

IV. OBTAINED RESULTS

To measure the impact of the usage of our framework two surveys were done. The first focused on the usability of the systems and the second on pedagogical issues. Additionally, we compared the grades obtained on previous semesters with the ones obtained in this current course, as well as de dropout rates. The framework was evaluated through the same principles established by the ISO Norm 9241- Part 10 - Dialogue Systems, based on the ten usability heuristics, proposed by Nielsen [50], to measure the general usability and the Nokelainen method was used for evaluating the pedagogic usability criteria [51].

This first set of questions, inspired by Nielsen's method, was applied to 15 students distributed in three teams. They evaluated the tool through general usability criteria, and the results were very promising. Namely 73.33% of the students (11 out of 15) were satisfied with the framework, 20.00% were partially satisfied (3 out of 15), and only one student was not satisfied (6.67%). The main reason for this negative result was that this student preferred classical classes with less practical exercises and claimed that the framework did not work as specified. In a next round, we plan to check along the course (in the first third of it) if all technical aspects of the framework are well understood and working properly.

Majority of the students claimed that the practical usage of the robot was a positive factor. They related that it was easier to understand some concepts because they could see a "response" for their abstract programs. They also affirmed that "playing" with the robot was one of the reasons to keep them motivated along the semester. One issue on the robot was the frequent battery recharge needed what sometimes caused delays on the execution of the practical experiments.

Another set of questions that used the Nokelainen's method was focused on the pedagogic usability. The goal was to verify if the tool along with the methodology was able to help students in the learning process. In other words, to verify if the proposed framework reached the expectation of the student in the classroom. Once again, the evaluation was very positive, 80.00% of the students were satisfied, 13,33% were partially satisfied, and 6,67% were unsatisfied with the tool and the methodology used.

The positive results obtained with this second questionnaire could be confirmed when we compared the average grades of this semester with previous ones. They raised considerably (about 15% better), but one can claim that different evaluation methods were used. Unfortunately, there is nothing to do with the existing results, in fact, we need to formulate a consistent evaluation methodology and follow the students' grades from now on.

What we can affirm is that the dropout rate, as well as the fail rate, reduced when compared to the same course in previous terms. The dropout obtained in this course was 18% lower than the known average while the fail rate reduced 12%. These results also must be followed up in the next semesters, and a set of questions about these two topics shall be applied.

V. CONCLUSION

This paper described the proposal of a Didactic Framework, whose objective is the teaching of the C programming language utilizing software tools, a mobile robot, and a cooperative/competitive learning method. The proposed methodology brings the students together in teams along one semester, which compete with other teams by trying to solve problems proposed by the lecturers. The proposed actions must be done in collaboration, and every activity resulted in points to the team.

Participating students were supposed to develop a software program to control a mobile robot. They delivered products developed by themselves (individually) and in collaboration with other students belonging to the same team. Every activity generates a score for them. In the end, one competition took place among the teams. In this proposal, all participants earned points, regardless of losing or winning the competition.

The available resources in our proposed didactic framework can increase and be adapted to the lecturers' goals along a specific course, what makes the tool interactive and flexible. The use of the mobile robot has proven to be a very effective way to improve learning. Finally, the competition at the end of the semester was a very interesting motivational instrument as well.

Next steps are related to finishing the deployment of the Didactic Framework. The final interface will be designed using concepts of usability for education, and it is supposed to increase the benefits of the teaching-learning process. A major survey will be done seeking feedback from students and lecturers through qualitative and quantitative assessments.

The Didactic Framework provides a promising approach to improve the learning of abstract concepts as well as improve the motivation of the participating students. Care must be taken during the application of the proposed methodology, and a balance between competition and cooperation should be achieved to increase the learning result.

Future works include repeating the experiment with new classes, integrate other programming languages, and insert new tools that consider web and mobile applications. We will also follow up the results and investigate the impact of the framework on the learning process closely.

ACKNOWLEDGMENT

We would like to thank you all participating students for their valuable suggestions. Thank you to all personnel of the funding agencies and from the Ambient Intelligence Laboratory from CETELI/UFAM.

REFERENCES

[1] K. Kori, M. Pedaste, H. Altin, E. Tõnisson, and T. Palts, "Factors That Influence Students' Motivation to Start and to Continue Studying Information Technology in Estonia," IEEE Transactions on Education, vol. 59, issue 4, pp. 255 – 262, 2016.

[2] H. Jonsson, "Motivating and preparing first-year students in computer and engineering science," 2013 IEEE Frontiers in Education Conference (FIE), pp. 1096 – 1102, 2013.

[3] E. D. Canedo, G. A. Santos, and S. A. A. de Freitas, "Analysis of the teaching-learning methodology adopted in the introduction to computer science classes," 2017 IEEE Frontiers in Education Conference (FIE), pp. 1 – 8, 2017.

[4] K. Kori, M. Pedaste, E. Tõnisson, T. Palts, H. Altin, R. Rantsus, R. Sell, K. Murtazin, and T. Rüütmann, "First-year dropout in ICT studies," 2015 IEEE Global Engineering Education Conference (EDUCON), pp. 437 – 445, 2015.

[5] M. Vesisenaho, H. Puhakka, J. Silvonen, E. Sutinen, M. Vanhalakka-Ruoho, P. Voutilainen, and L. Penttinen, "Need for study and career counseling in computer science," 39th IEEE Frontiers in Education Conference, pp. 1 – 6, 2009.

[6] D. Ktoridou, and E. Epaminonda, "Measuring the compatibility between engineering students' personality types and major of study: A first step towards preventing engineering education dropouts." 2014 IEEE Global Engineering Education Conference (EDUCON), pp. 192 – 195, 2014.

[7] T. Jungmann, and P. Ossenberg, "Recruiting the right engineering students," 2013 IEEE Global Engineering Education Conference (EDUCON), pp. 834 – 839, 2013.

[8] S. Kujala, A. Lehtovuori, and M. A. Honkala, "Promoting positive start of electrical engineering studies — A teacher's perspective," 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, pp. 1 – 8, 2014.

[9] J. Pelech, and G. Pieper, "The Comprehensive Handbook of Constructivist Teaching: From Theory to Practice," Information Age Publishing, 228 p., 2010.

[10] J. K. Mohapatra, M. Mahapatra, and B. K. Parida, "Constructivism: The New Paradigm (From Theory to Practice)," Atlantic Publishers and Distributors Pvt Ltd, 379 p., 2016.

[11] J. Piaget, "Psychology, and Epistemology: Towards a Theory of Knowledge," New York, Grossman, 1971.

[12] M. Montessori, "The Montessori Method," CreateSpace Independent Publishing Platform, 254 p., 2012.

[13] P. P. Lillard, "Montessori Today: A Comprehensive Approach to Education from Birth to Adulthood," Schocken Publishing, 240 p., 1996.

[14] American Montessori Society, Introduction to Montessori Method, https://amshq.org/Montessori-Education/Introduction-to-Montessori, (accessed June 2018).

[15] J. P. Diggins, "The Promise of Pragmatism: Modernism and the Crisis of Knowledge and Authority," University of Chicago Press, 528 p., 1995.

[16] G. Gutek, "Philosophical, Ideological, and Theoretical Perspectives On Education," New Jersey: Pearson, p. 76, 2014.

[17] V. F. Lucena, A. Brito, and P. Gohner, "A Germany-Brazil Experience Report on Teaching Software Engineering for Electrical Engineering Undergraduate Students," 19th Conference on Software Engineering Education & Training (CSEET'06), pp. 69-76, April 2006.

[18] S. I. Ahamed, "Experiences in teaching an object-oriented design and data structure course," Proc. ITCC 2003. International Conference on Information Technology: Coding and Computing, pp. 48 – 52, 2003.

[19] A. Robins, J. Roundtree, and N. Roundtree, "Learning and Teaching Programming: A Review and Discussion," Computer Science Education, Vol. 13 (2), pp.137-173, 2003.

[20] A. Vizcaino, and B. du Boulay, "Using a simulated student to repair difficulties in collaborative learning," International Conference on Computers in Education, Proceedings, pp. 349 - 353 vol.1, 2002.

[21] F. Y. Yu, L. J. Chang, Y. H. Liu, and T. W. Chan, "Learning preferences towards computerized competitive modes," Journal of Computer Assisted Learning, Vol.18(3), pp.341-350, September 2002.

[22] K. Sanders, J. Boustedt, A. Eckerdal, R. McCartney, and C. Zander, "Folk Pedagogy: Nobody Doesn't Like Active Learning," ICER '17: Proceedings of the 2017 ACM Conference on International Computing Education Research, pp. 145-154, August 2017.

[23] C. A. Jara, F. A. Candelas, F. Torres, S. Dormido, and F.Esquembre, "Synchronous collaboration of virtual and remote laboratories," Computer Applications in Engineering Education, vol. 20, pp. 124-136, 2012.

[24] P. Moreno-Ger, I. Martinez-Ortiz, V. Gilmartin, and R. Ballesteros, "Trivialcv: Competitive activities for the classroom integrated in a Moodle virtual campus," IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 8, no. 1, pp. 31–38, Feb 2013.

[25] B. S. Jong, C. H. Lai, Ye. T. Hsia, T. W. Lin, and C. Y. Lu, "Using Game-Based Cooperative Learning to Improve Learning Motivation: A Study of Online Game Use in an Operating Systems Course," IEEE Transactions on Education, vol. 56, No. 2, pp. 183-190, May 2013.

[26] C. A. Dugas, "No Computer? No Problem! Active and Cooperative Learning in an Introductory Computer Science Course," In 2008 Frontiers in Education Conference Proceedings, pp. 1-4, October 2008.

[27] H. Bajwa, and P. Mulcahy-Ernst, "Redesigning Teaching Approaches for Undergraduate Engineering Classrooms," In 2012 Integrated STEM Education Conference, pp. 1-4, March 2012.

[28] OECD Better Life Index.; Available in http://www.oecdbetterlifeindex.org/topics/education/, [acessed 2018 April 17]

[29] S. Mishra, C. Romanowski. R. Raj, T. Howles, and J. Schneider, "A Curricular Framework for Critical Infrastructure Protection Education for Engineering, Technology and Computing Majors." In 2013 Frontiers in Education Conference Proceedings, October 2013.

[30] S. Krithivasan, S. Shandilya, K. Arya, K. Lala, P. Manavar, S. Patil, and S. Jain, "Learning by Competing and Competing by Learning: Experience from the e-Yantra Robotics Competition," In 2014 Frontiers in Education Conference Proceedings, pp. 1-8, October 2014.

[31] V. Plestina, H. Turic, and V. Papic, "Constructive Education Approach: Robot Soccer," In Proceedings of the 29th International Conference on Information Technology Interfaces, pp. 425-430, 2007.

[32] D. J. Pack, and A. R. Klayton, "Education through Robotics at the United States Air Force Academy," 2006 World Automation Congress, pp. 1 – 6, 2006.

[33] H. C. S. Thomazinho, A. L'Erario, and J. A. Fabri, "Teaching software maintenance with ludic techniques supported by robotics," 2017 IEEE Frontiers in Education Conference (FIE), pp. 1 – 8, 2017.

[34] L. Major, T. Kyriacou, and O. P. Brereton, "Systematic literature review: Teaching novices programming using robots," 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011), pp. 21 – 30, 2011.

[35] L. D. O. Maia, V. J. da Silva, R. E. V. de S. Rosa, J. P. Queiroz-Neto, and V. F. de Lucena, "An experience to use robotics to improve Computer Science learning," 2009 39th IEEE Frontiers in Education Conference, pp. 1 – 6, 2009.

[36] J. Schreurs, and A. Al-Huneidi, "Development of a learner-centered learning process for a course." In Proceedings of the 14th International Conference on Interactive Collaborative Learning, pp. 256-263, 2011.

[37] R. Raval, A. Maskus, B. Saltmiras, M. Dunn, P. J. Hawrylak, and J. Hale, "Competitive Learning Environment for Cyber-Physical System Security Experimentation," 2018 1st International Conference on Data Intelligence and Security (ICDIS), pp. 211 – 218, 2018.

[38] G. Braught, T. Wahls, and L. M. Eby, "The Case for Pair Programming in the Computer Science Classroom," February 2011 ACM Transactions on Computing Education (TOCE), vol. 11, issue 1, pp. 1-21, February 2011.

[39] M. Huggard, F. Boland, and C. Mc Goldrick, "Using Cooperative Learning to Enhance Critical Reflection." In 2014 Frontiers in Education Conference Proceedings, pp. 1-8, October 2014.

[40] R. Swearer, "Industry-Backed Competitions: Helping Today's Students Prepare for Tomorrow's Careers," IEEE Computer Magazine, vol. 50, issue 7, pp. 23 – 25, 2017.

[41] S. Willis, G. Byrd, and B. D. Johnson, "Challenge-Based Learning," IEEE Computer Magazine, vol. 50, issue 7, pp. 13 – 16, 2017.

[42] H. Jin, “A hybrid instructional design model for the combination of motivation theory and constructivism,” In Proceedings of the 4th International Conference on Computer Science & Education, pp. 1652-1656, 2009.

[43] J. R. Buck, and K. E. Wage, “ Active and cooperative learning in signal processing courses,” IEEE Signal Processing Magazine, Vol. 22, No. 2, pp. 76-81, 2005.

[44] R. Vickers, G. Cooper, J. Field, M. Thayne, R. Adams, and M. Lochrie, “Social media and collaborative learning: hello Scholr,” AcademicMindTrek '14: Proceedings of the 18th International Academic MindTrek Conference: Media Business, Management, Content & Services, Publisher: ACM, pp. 103-109, November 2014.

[45] G. Stahl, “Perspectives on Collaborative Learning,” Group Cognition: Computer Support for Building Collaborative Knowledge, MIT Press eBook Chapters, pp. 119 – 153, 2006.

[46] M. Blanco, C. Gonzalez, A. Sanchez-Lite, and M. A. Sebastian, “A Practical Evaluation of a Collaborative Learning Method for Engineering Project Subjects,” IEEE Access, vol. 5, pp. 19363 – 19372, 2017.

[47] M. Marques, S. F. Ochoa, M. C. Bastarrica, and F. J. Gutierrez, “Enhancing the Student Learning Experience in Software Engineering Project Courses,” IEEE Transactions on Education, vol. 61, issue 1, pp. 63 – 73, 2018.

[48] C. Tingley, G. Serwin, R. Tuli, J. K. Posner, C. Lisy, L. Bresee; K. Viadro, and C. Morimoto, “Turning an International Competition into a Valuable Learning Experience,” IEEE Computer Magazine, vol. 51, issue 1, pp. 76 – 79, 2018.

[49] Festo Didactic, available in http://www.festo-didactic.com/int-en/learning-systems/education-and-research-robots-robotino, [acessed 2018 April 17]

[50] J. Nielsen, “Usability Engineering,” Chapter 3 – Generations of User Interfaces, 223 p - pp.49-69, 1993.

[51] J. Li, “The role of education in the learning process,” In 2010 International Conference on Artificial Intelligence and Education (ICAIE), pp. 85-88, 2010.