IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 7 Issue 11, November 2020
ISSN (Online) 2348 – 7968 | Impact Factor (2020) – 6.72
www.ijiset.com

The abstract concepts and SPHERO

Alberto YÁÑEZ-CASTILLO[1], Ana LAUREANO-CRUCES[1,2], Gustavo DE LA CRUZ-MARTÍNEZ[3], Javier RAMÍREZ-RODRÍGUEZ[2]

[1]Posgrado de Diseño y Visualización de la Información - Universidad Autónoma Metropolitana – Azcapotzalco
[2] Departamento de Sistemas - Universidad Autónoma Metropolitana - Azcapotzalco
[3] Instituto de Ciencias Aplicadas y Tecnología – Universidad Nacional Autónoma de México
alberto_uam@hotmail.com,
clc,jararo@{azc.uam.mx}, gustavo.delacruz@icat.unam.mx

Abstract

In this work, a series of characteristics are touched upon that allow us to understand the problems that structured programming students face. A novel methodology is proposed that allows the visualization of the abstract concepts that constitute it. In this case, Forbus's theory and its qualitative reasoning are used, through which it is intended to bring the abstract concept to a physical plane. With the arrival of technology and computers in the early 1990s, research into the use of these new technologies began in order to reformulate the traditional form of the teaching-learning process; the case study refers to the visualization of abstract concepts. To achieve this, articulated mini - theories are developed on mastering the abstract concepts of structured programming, using the mental models of the expert in and the theory of qualitative reasoning. By implementing daily life examples applied to teaching resources in the case study, the robot *SPHERO* has been chosen. A technological toy that can be programmed (i.e., controlled) by means of a computer or mobile device using the *SPHERO Eduvisual* programming application. The purpose is to achieve an interactive experience, allowing a connection between the real and abstract of programming in a visual and tangible way. Following, the abstract concepts of programming are defined in detail, and the possible reasons or factors that may intervene in their comprehensions. The proposal of this work is explained through the theory of qualitative reasoning.

Keywords: *structured programming, abstract concepts, mini-languages, qualitative reasoning, SPHERO.*

1. Introduction

Within the area of computer science and engineering exists structured programming, which is regularly a course pivot for engineering and computer related careers. The **programming term consists of comprehend a problem to solve it by means of a computer,** in parallel it is necessary to comprehend how the computer will be able to arrive at the solution [1]. For others, it is the art of building programs by means of a set of comprehensive instructions that are the solution to a problem executed by the computer [2]. In other words, **programming is a complex mental process; it is not about learning a procedure or memorizing knowledge, but rather solving a certain problem using a computer and a series of understandable instructions.**

To learn structured programming is required at least **mastery of abstract concepts** relating to the same programming and certain basic capabilities such as: logical thinking, mathematical logic and reasoning. Also, the student must have a good amount of time and mental effort [3, p.308].

The basic abstract concepts that the student of structured programming must comprehend are: **variable, sequencing, simple selection, conditional iteration, and fixed iteration.** Some explanations or definitions given in the course for these concepts are:

- **Variable**: this must be thought of as a memory position, where its content can be varied by means of assignment instructions [4]. It is also a symbolic name of the starting address of a group of memory cells that contains a value [3]

- **Sequencing**: *execute the indicated actions, one after the other* [3, p.313], that is, following a disciplined sequence of successive steps in order to describe, model, or solve a situation.

- **Simple selection:** a logical expression (condition) is evaluated which can have the value of *true or false*, once the condition has been evaluated and knowing the value obtained, action *one* is executed; if the value is true or action *two*; for the value of false. Emphasizing; that action *one* or action *two* are represented by a set of sequential actions.

- **Conditional Iteration:** a logical expression (condition) is evaluated which can have the value of *true or false*, similar case to the simple selection, but this time if the value turns out *to be true*, one or more actions are executed repeatedly while the condition value remains *true*; If the value is *false*, the execution of the action or actions ends or the cycle may never start because from the beginning the value turned out to be *false* [3].

- **Fixed iteration**: in this case the action (s) is executed repeatedly for a certain number of times. Another way to explain it is considering that the value of the number is the condition to fulfill to stop the fixed iteration.

These basic abstract concepts have a one-to-one relationship with control structures: 1) sequencing, 2) simple selection, 3) conditional iteration, and 4) fixed. With them it is possible to write any program. All control structures operate on conditional statements, that is, a logical expression with an informative value that can be true or false. Once the result is obtained, the appropriate actions are produced.

2. Reasons and factors that may intervene in the comprehension of abstract concepts

In our standpoint, *that comprehension of abstract concepts is important for any engineering student.* The case study refers to programming *firstly*, because it is part of the curriculum of all engineering careers, *secondly*, because today's world has technological needs that someone must meet, and *lastly*, because programming contributes to develop computational thinking in students, useful to think recursively, solve problems, and think in parallel, among others [5]. Which is useful even to solve the problems of daily life?

Why do some programming students find it difficult to comprehend abstract programming concepts and control structures?
To try to answer the previous question, the judgment of various authors will be exposed to the problems they have observed in programming students.

According to Ruiz Velasco [1], mathematics and the computer language (programming) are of great importance because of the application they may have in any domain or activity, that is, *they help people to think about their lives, to organize their knowledge and to develop socially, emotionally and intellectually* [5, p.2]; however, students find obstacles to master certain abstract concepts, probably caused because *there is no relationship between the use and manipulation of these concepts with everyday situations* [5, p.2].

Carlos Chesñevar [6], assures that many students from the secondary level graduate with poor training and this makes *it difficult for them to organize new concepts in an orderly manner*. Furthermore, programming in some programming language requires considerable effort, as well as *various competencies and skills, which basically involve the ability to manipulate a set of interrelated abstractions* [6].

In Insuasti [7], reasons why the student fails to learn programming are exposed: 1) the concepts of programming, 2) the cognitive load involved in programming learning, and 3) the lack of proper cognitive skills to solve problems (p. 236); These reasons have been presented in research published through the ACM (*Association for Computing Machinery*).

Showing up next, some of these reasons, as identified by their authors. Regarding the cognitive programming task, Baldwin and Kuljis [8] identified that learning programming requires **cognitive skills** such as: 1) planning, 2) reasoning, and 3) problem solving (cited in Insuasti, [8]). And according to Insuasti [7] this demand also extends to the development of

thinking skills such as: 1) the capacity of attraction of the study object [9], [2], 2) the ease of analysis, and 3) the skill for synthesis. Such skills are a complement to problem solving. A general rule implies; if these skills have not been developed, they could be part of the difficulty that students have with the task of programming.

Likewise, there are factors such as: 1) the student's inability to know what happens with the execution of the instructions, calculations and results that the program performs, 2) the lack of motivation to program, and 3) the difficulty in comprehend the compound logic [7].

Insuasti [7], complements the above with something very important, the *difficulty faced by programming students is the inability to imagine and comprehend abstract terms that have no equivalent in real life*, for example, how could they imagine or relate a variable with a real-life object? Or perhaps a metaphor is not necessary to relate abstract programming concepts to some real-life object?

Figure 1, summarizes factors or reasons that may intervene in the lack of comprehension of abstract concepts in programming.

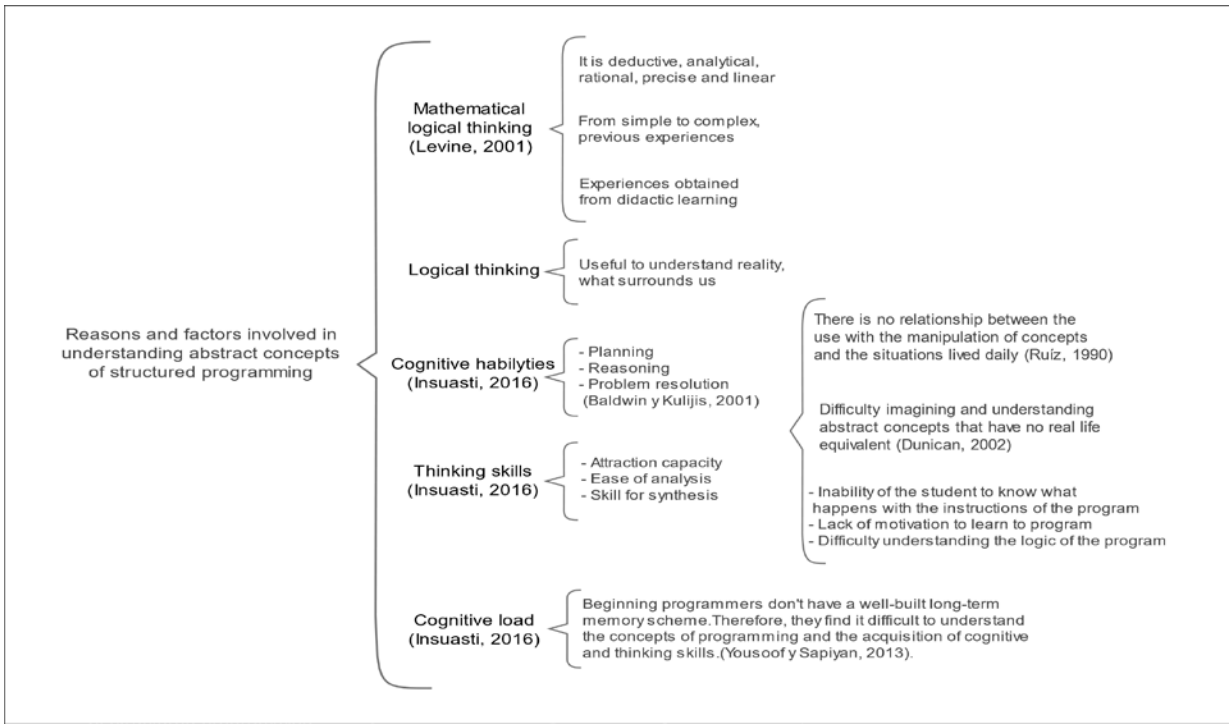

Figure 1. ***Reasons and Factors that can intervene in the understanding of abstract programming concepts (Source: own elaboration)***

3. Some proposals to learn to program

The most common and current proposals to teach programming to students derive in two aspects, in some proposals these are merged into one. The first aspect is the use of mini-languages, and the second aspect involves physical objects, robots or devices custom designed with their own programming language. In the case of study, the second aspect is used.

The list of robots that can be programmed through a visual programming environment is endless, many of them are quite high cost and therefore difficult to access for students, but one that is quite practical and economical is SPHERO mini which shown in Figure 2. A small sphere-shaped robot connects via Bluetooth to mobile devices with Android, iOS, Kindle operating systems and to computers with Bluetooth and Mac or Windows operating systems [10].

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 7 Issue 11, November 2020
ISSN (Online) 2348 – 7968 | Impact Factor (2020) – 6.72
www.ijiset.com

Figure 2. SPHERO mini [19].

The *SPHERO Edu* app is available for each of the operating systems, which allows building programs and interacting with the robot through a visual programming environment. This application has different programming blocks, including control structures: 1) simple selection, 2) fixed iteration, and 3) conditional, it also has an iteration variant 4) repeat indefinitely, that is, it has no condition to execute the action or not. They are shown in Figure 3.

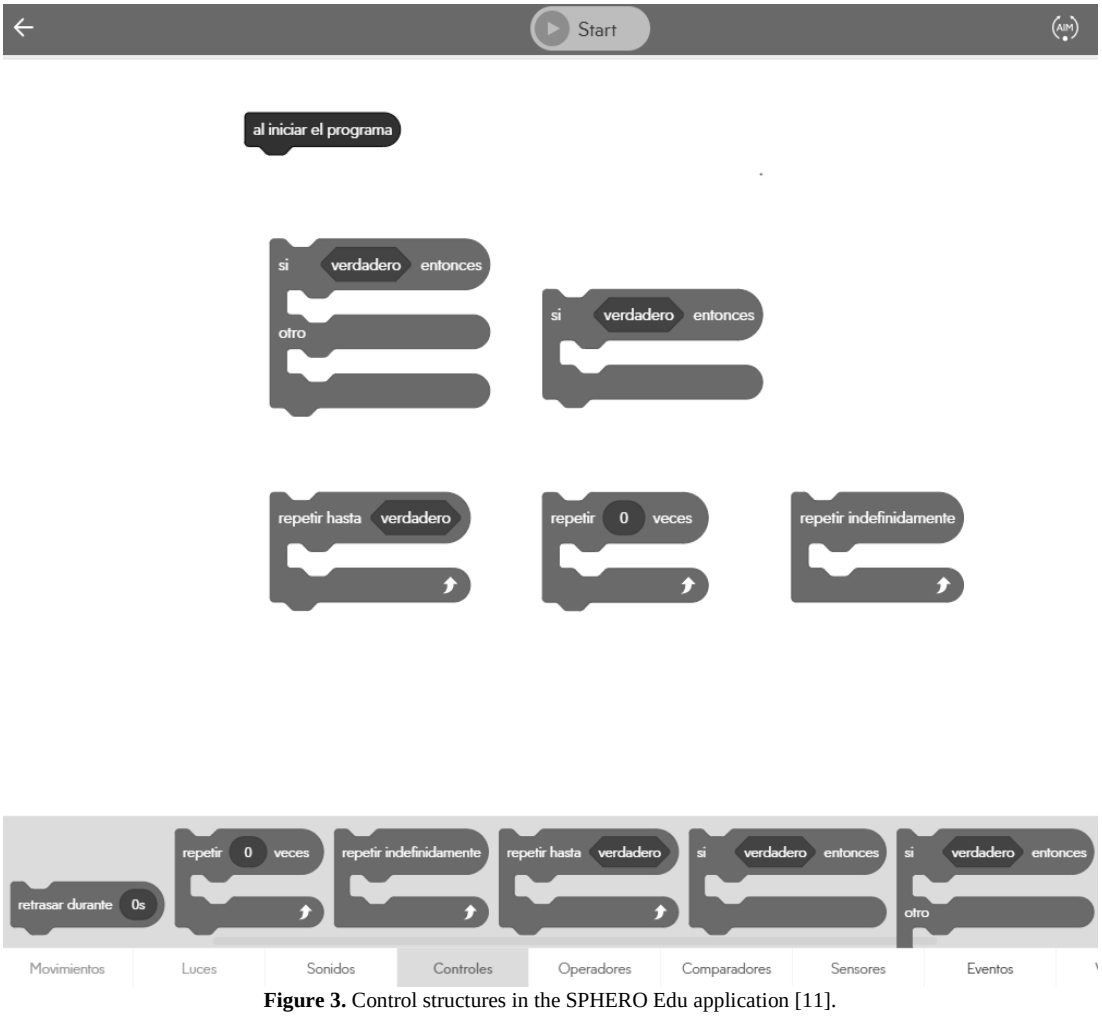

Figure 3. Control structures in the SPHERO Edu application [11].

4. Qualitative reasoning

Within artificial intelligence there is (a sub-discipline), the qualitative physics its objective is the description of physical systems. They represent and explain their behavior based on *common sense* used by humans to qualitatively analyze the environment and on the scientific knowledge used by engineers [12]. The objective is: *to predict, describe and explain the behavior of physical systems, without having to resort to mathematics*, that is, to make it simpler and in qualitative terms. The above is part of qualitative reasoning, such reasoning is widely used by scientists and engineers [13].

Qualitative reasoning has been considered a domain of artificial intelligence since 1984 with the publication of a special issue of the magazine Artificial Intelligence. According to [14], qualitative reasoning creates representations for the continuous aspects of the world, such as space, time, and quantity. This reasoning is based on two characteristics: 1) people draw useful and ingenious conclusions about the physical world without the use of differential equations, that is, in daily life people may discover what is happening or happening around them and how to change it (or affect), working with less data or insufficient information, and 2) apparently, both scientists and engineers use qualitative reasoning to initially comprehend a physical problem [14].

Objects that change with respect to time characterize a **process**. The events that cause an object to change are: 1) movement, 2) colliding with each other, 3) getting hot, 4) cooling, 5) stretching, and 6) compressing, among others. In formal physics, these processes are characterized by differential equations that describe how the material parameters of objects change over time; but what happens during a physical process, we can usually find out or conclude with insufficient information in hand. For example, if we heat water in a closed container, the water boils after a certain time, and we know that if the container continues to heat; the water will continue to boil until the container will most likely explode when the water evaporates. So, to comprehend physical common-sense reasoning, you need to understand how to reason qualitatively about **processes**, for example, when they will occur, when they will stop, and the effects of such processes. For this we use the qualitative theory of the process [15]. This author proposes seven types of reasoning, these are used as a means (tasks) to articulate the procedures for comprehension a certain concept in a specific domain.

4.1 Reasoning tasks

Within the qualitative theory of the process different tasks (or styles) of reasoning are described that may be appropriate for the comprehension of different kinds of abstract phenomena; it is not mandatory to use all; the most appropriate should be used according to experts in the domain. The different types of reasoning proposed by Forbus [15] are specified below:

- **Determining activity**: deduce what is happening in a situation, during a particular moment. Provides direct answers to a class of questions - *what is happening here*; being a basic operation in some cases for the other types of reasoning.

- **Prediction:** deduce what will happen in the future of some situation. By working with incomplete information, only descriptions of *possible futures can be obtained, rather than a single future.*

- **Postdiction**: deduce how a particular state of affairs could have arisen. This type of reasoning is more difficult than prediction due to the need to postulate individual hypotheses that explain the possible reasons for the fact; being more difficult to use due to its difficulty in chaining (backwards) what happened, since there is imprecision in the information [16].

- **Skeptical analysis:** determine if the description of a physical situation is consistent.

- **Measurement interpretation:** given a partial description of the variables that create a situation and some observations of its behavior, infer if there are other variables and what else could happen.

- **Experimental planning**: given a knowledge of what can be observed and what can be manipulated, plan actions that produce more information about the situation.

- **Causal reasoning**: consider a description of the behavior that generates changes to particular variables of the situation. The cause-effect relationship (causality) can be a tool to give credit to the hypothesis about an observed behavior [17], [18], [19].

5. The proposal of this work

The main idea is to transform an abstract concept into a physical event that allows it to be perceived through the senses. Mini develop - theories to divide the domain of abstract concepts to facilitate the way to deal with the context, that is, **people are able to answer questions and make predictions about the fictional worlds, even knowing that they do not exist** [20]**.**

The qualitative reasoning and its realization through the mini - theories developed for each concept, occupy a prominent place among the possible aids; in order to facilitate the description of processes and solution of certain problems in a specific domain [3].

The micro - theories will be designed for the different abstract concepts, creating scenes with the help of SPHERO; where the necessary properties are discovered for the understanding of the concepts involved. And as developed by [19] in their research work, different stories are implemented according to each of the reasoning tasks (or styles) and depending on the abstract concept to be addressed.

5.1 Mini - Theory

Micro-theory for the concept of simple selection

Understanding the control structure concepts: 1) **simple selection,** 2) **fixed iteration.**

1) Simple selection

Objective: define simple selection based on the selection of one path or another, the selection is decided by the result of a true logical expression, the action will be executed by the *SPHERO* robot in a fictional micro world.

Marco: the *SPHERO* robot goes on a path in a fictional micro - world, finds clues (in this case data to evaluate a logical expression) on his path, but when he reaches a certain point he is at a crossroads, *SPHERO* must choose the path of the true logical expression, for this it will require making use of the clues and evaluate logical expressions.

Reasoning tasks

- **Determining activity:** deduce what happens in a situation and at a certain moment. Provide direct answers to a class of questions, such as: what is going on here?

For this reasoning task, the student must deduce what happens when the robot reaches the crossroads, that is, what *SPHERO* does to go forward, in addition to answer the following questions:

- ✓ What about the clues that *SPHERO* found?
- ✓ Which path corresponds to the *true* logical expression?
- ✓ Why does the robot choose the path corresponding to the *true* logical expression?
- ✓ What does the robot do with the logical expressions? And in his own words define the concept of simple selection.

2) Fixed iteration

Objective: define fixed iteration based on the repetition of actions of the Sphero robot, defined by a fixed number.

Marco: in a micro - world, the Sphero robot finds a number that allows it to know how many times it must repeat a set of actions; the actions will be linked to different arithmetic problems. This set of actions can be empty.

Reasoning tasks

- **Determining activity:** know what is going to happen at the end of the cycle. Taking into consideration the set of actions and the particular variables affected within the cycle.

In this case, a description of the behavior that generates changes to particular variables that are within the cycle must be considered; as well as the variable that controls the cycle.

6. Conclusions

So far, a study has been conducted from the pedagogical perspective in which the difficulties that are presented to programming students are discovered. This has prompted the scientific community to discover new forms of the teaching-learning process. Using new technologies that allow us to venture into different types of visualization.

This work is probably one of the first to extrapolate qualitative process theory to the concepts of control structures that are part of what we call abstract programming concepts. In the study case, we consider that we have two channels through which we are visually connecting: 1) the code that is displayed so that the SPHERO achieves its activity, which implies, 2) the activity of SPHERO in the physical environment.

The proposal of this research work is based on the cognitive process that underlies programming from here, the emphasis on students comprehend abstract concepts and not learning to program. Once the concepts underlying the programming skill are understood, the latter will be given. We have made the first examples with SPHERO; that can be viewed in real time in the classroom or conducted through online classes.

Acknowledgments

This work is part of the research project conducted by José Alberto Yáñez-Castillo to obtain the Ph.D. degree in Diseño y Visualización de la Información de la Universidad Autónoma Metropolitana - Azcapotzalco; as well is part of the divisional project *Design of intelligent interfaces for simulating the behavior of living or animate organisms: information visualization* section; from the same University available in http://kali.azc.uam.mx/clc/.

References

[1] Ruiz-Velasco E., “La informática como medio de enseñanza y objeto de aprendizaje”, Perfiles educativos, México, pp. 37–43, 1990.
[2] Yañez-Castillo J. A., Laureano-Cruces A. L., Garmendia-Ramírez G. I., “Diseño emocional de elementos interactivos visuales: una perspectiva para mejorar la interacción de los inmigrantes digitales con la computadora personal”, Tecnología & Diseño, vol. 6, no. 8, Dec. 2017.
[3] Levine G., “Computación y programación moderna. Perspectiva integral de la informática”, México, Pearson Educación de México, 2001.
[4] Sánchez-Guerrero M. de L., “Sistemas de Aprendizaje Inteligente con Objetos de Aprendizaje (ProgEst)”, MSc. Degree in Computer Sciences thesis, Universidad Autónoma Metropolitana, 2009, Available: http://kali.azc.uam.mx/clc/02_publicaciones/tesis_dirigidas/tesis_final_lsgnov09.pdf.

[5] Sáez J. M. , Cózar R., “Pensamiento computacional y programación visual por bloques en el aula de Primaria”, Educar, vol. 53, no. 1, pp. 129–146, 2017.
[6] Chesñevar C. I., “Utilización de los mapas conceptuales en la enseñanza de la programación”, vol. 3, p. 11, 2000.
[7] Insuasti J., “Problemas de enseñanza y aprendizaje de los fundamentos de programación”, Revista Educación y Desarrollo Social, vol. 10, no. 2, pp. 12011–5318, 2016.
[8] Baldwin L., and Kuljis J., “Learning Programming Using Program Visualization Techniques”, 2001.
[9] Laureano-Cruces A. L., Sanchez-Guerrero M. de L., Velasco-Santos P., Mora-Torres M., “The interface: an object that is hated and loved”, E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education. Association for the Advancement of Computing in Education (AACE), San Diego, CA, Vancouver, British Columbia, Canada, pp. 380–388, 17-Oct-2017.
[10] “Sphero Mini [Online]”, Available: https://www.sphero.com/sphero-mini, [Accessed: 06-Dec-2019].
[11] Trejos-Buriticá O. I., “Consideraciones sobre la evolución del pensamiento a partir de los paradigmas de programación de computadores”, Revista Tecnura, vol. 16, no. 32, p. 68, 2012.
[12] Bobrow D. G., “Qualitative reasoning about physical systems: an introduction”, Artificial Intelligence, vol. 24, no. 1–3, pp. 1–5, 1984.
[13] Laureano-Cruces A. L., Sanchez-Guerrero M. de L., Velasco-Santos P., Mora-Torres M., “The interface: an object that is hated and loved”, E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education. Association for the Advancement of Computing in Education (AACE), San Diego, CA, Vancouver, British Columbia, Canada, pp. 380–388, 17-Oct-2017.
[14] Forbus K. D., “Qualitative reasoning”, in Computer Science Handbook, Second Edition, A. B. Tucker, Ed. Evanston, Illinois: Chapman and Hall/CRC, 1996, p. 29.
[15] Forbus K. D., “Qualitative Process Theory”, Artificial Intelligence, vol. 24, pp. 85–168, 1984.
[16] Aliseda A., “Seeking explanations: abduction in logic, philosophy of science and artificial intelligence”, ILLC-Publications, Amsterdan, 1997.
[17] Laureano-Cruces A. L., Barceló-Aspeitia A. A., “Formal verification of multi-agent systems behaviour emerging from cognitive task analysis”, J. Exp. Theor, Artif. Intell., vol. 15, no. 4, pp. 407–431, 2003.
[18] Laureano-Cruces A. L., De Arriaga-Gómez F., “Reactive Agent Design for Intelligent Tutoring Systems”, Cybernetics and Systems, vol. 31, no. 1, pp. 1–47, 2000.
[19] Laureano-Cruces A. L., Terán-Gilmore A., De Arriaga F, “A Learning Cognitive Model Based on a Didactic Cognitive Approach: The Case of Single-Degree-of-Freedom Systems”, Computer Applications in Engineering Education, 2004, vol. 12, no. 3, pp. 152-164.
[20] Forbus K. D., “Modeling amidst the Microtheories”, in Proceedings of the 24th International Workshop on Qualitative Reasoning, 2010, pp. 112–115.

J. Alberto YAÑEZ-CASTILLO, studied the Bachelor's Degree in Computer Engineering, a Master’s degree in Diseño y Visulaización de la Información at Universidad Autónoma Metropolitana. He has taught various courses and workshops on programming for interfaces cognitive design. His main areas of interest are: Affective Computing, Cognitive Emotions, Behavioral Analysis, Artificial Consciousness and Interfaces design.

Ana LAUREANO-CRUCES, obtained a Bachelor’s degree in Civil Engineering, a Master’s degree in Computer Sciences, and a Ph.D. degree in Science at UNAM. She has been a full-time professor at Universidad Autónoma Metropolitana - Azcapotzalco since 1987. She was a guest researcher between 1) 1998 and 2000, at the Departamento de Sistemas del Instituto de Automática Industrial del Consejo Superior de Ciencia y Tecnología de Madrid (SPAIN), 2) September 2011-April 2013 at Université d'Avignon et des Pays de Vaucluse (UAPV, FRANCE). Her main areas of interest are: Expert Systems and Intelligent Systems applied to the teaching-learning process, Affective Computing, Cognitive Emotions, Reactive Agents, Pedagogical Agents, Multi-Agent Architectures, Behavioral Analysis, Adaptive Control, Cognitive Engineering, Knowledge Representation, and Artificial Consciousness.

Gustavo DE LA CRUZ-MARTÍNEZ, Ph.D. in Computer Science at UNAM. Founding member of the project of the classroom of the future. Professor at the Faculty of Sciences of the UNAM since 2002 and full-time Academic at the Institute of Applied Sciences and Technology of the UNAM since 2005. His areas of interest are human-computer interaction, user modeling and user experience.His current lines of research are: Methodologies for the design and evaluation of user experience, Cognitive user modeling, Interactive spaces and non-formal education.

Javier RAMÍREZ-RODRÍGUEZ, is full Professor in the Departamento de Sistemas at Universidad Autónoma Metropolitana in Mexico City. Doctor in Mathematics from Complutense University of Madrid (Spain). He has held lecturing positions at Posgrado de Ingeniería, UNAM and visiting professor at Université d’Avignon et des Pays de Vaucluse, France. His research interests are in exact and heuristics methods, combinatorial optimization and soft computing.