Title: **Computational Thinking in Pre-University Blended Learning Classrooms**

Authors: **Xabier Basogain**
Escuela Superior de Ingeniería
The Basque Country University (EHU)
Bilbao, Spain
xabier.basogain@ehu.eus

Miguel Ángel. Olabe
Escuela Superior de Ingeniería
The Basque Country University (EHU)
Bilbao, Spain
miguelangel.olabe@ehu.eus

Juan Carlos Olabe
Electrical & Computer Engineering
Christian Brothers University, (CBU)
Memphis, TN, USA
jolabe@cbu.edu

Mauricio Javier Rico Lugo
Instituto Colombiano de Aprendizaje (INCAP)
Bogotá, Colombia
mauricio.rico@incap.edu.co

Computational Thinking in Pre-University Blended Learning Classrooms

Abstract—This article describes the implementation of various core elements of Computational Thinking in the classrooms of schools of Latin America and USA in two specific courses: PC-01 and ECE130. These courses were designed for students of primary and secondary education, as well as for students of high school as part of a dual enrollment program with a local university. Both courses introduce the core "concepts" and "processes" of Computational Thinking aided by the Scratch and Alice visual programming environments. The courses are designed to be facilitated by the classroom teacher with the support of a learning platform. This platform is supported by Moodle and it is configured to provide innovative pedagogical strategies based on emerging educational technologies. The first part of the article includes a comprehensive reflection on the concepts integrated under the term 'Computational Thinking.' This is followed by a discussion on the potential benefits of using a virtual learning environment in order to incorporate Computational Thinking in the classroom. The article includes a detailed description of syllabi and assessments (tests and peer to peer projects) of both courses, and it concludes with a comprehensive description of the impact of these courses on the educational institutions, teachers and students, of the Dominican Republic and USA, where the courses were implemented.

Keywords— Computational Thinking; Learning Technologies; Scratch; Alice; Educational Technology.

1. Introduction

After a period of transition in which the fundamental ideas promoted by Computational Thinking (CT) were studied and analyzed by all constituencies of public and private education, it appears that a substantial consensus is being forged which can shape future decisions and policies in K-12 education (Hubwieser, Armoni, Giannakos and Mittermeir, 2014). One important consequence of this confluence of ideas is the definition of a K-12 curricula with the integration of the core ideas of CT, out of which the particular classroom paradigms will be developed. In addition, it is universally agreed that programming, in its various forms, is an absolutely necessary mechanism for the implementation of fundamental concepts of CT as well as the best practices of CT education. Programming provides the three mechanisms that a language requires for the creation of complex systems: set of primitives; means of combination; and abstraction. Unlike natural languages, which by design are semi-structured and provide limited ability of multilevel abstraction, programming languages have been designed to fulfil the core ideas of CT. A mind equipped with the mechanisms of well-designed object oriented programming languages is prepared to house the principles of CT and is prepared to developed CT ideas.

The formal study of computational skills in primary and secondary schools has been recognized by many institutions and administrations. For example England, beginning during the academic year 2014-15, formally incorporated the study of computational thinking and computer programming as part of the curriculum of primary and secondary education, as described in the national curriculum of England: Study of Computer Program (Department for Education England, 2013).

The governments of the developed world see CT as the cornerstone of a technological society, and the governments of the developing world see CT as their best chance to close the gap of their education systems.

The Code.org organization (Code.org, 2012) promotes the idea that all students should have the opportunity to learn programming. This initiative has the support of important public figures of Microsoft, Facebook and the world of technology in general.

Programming environments such as Scratch, ScratchJr and Alice among others, play an essential role in this process. These programming environments allow the creation of programs that could be described as games or stories, or combination of interactive stories and games. Scratch, ScratchJr, and other graphic programming environments were designed specifically to address the developmental and learning needs of children (ages 5-7), and youth adults (ages 8-15), (Flannery, Silverman, Kazakoff, Bers, Bontá & Resnick, 2013; Resnick, Maloney, Monroy-Hernández, Rusk, Eastmond, Brennan & Kafai, 2009); Sáez López, González & Cano, 2016).

Alice (Alice, 2017) is a graphical programming language and environment integrated within a three dimensional world. It is an object oriented programming environment, where in addition to programming with graphic interlocking blocks, the programmer has immediate access to the java code that it is been created behind the scenes. Because Alice hides from the

user the syntactical complexity of java, and because it is integrated within a 3D environment, it provides young students an ideal environment to learn the fundamental programming concepts in the motivating context of creating movies and video games (Zhang, Liu, Ordónez de Pablos & She, 2014). These movies and games, in addition to providing a perfect platform to students to express their interest and creative ideas (Denner, Werner & Ortiz, 2012), offer a rich object oriented world where human cognitive primitives (objects with properties, behaviors and interfaces with other objects) optimally develop the fundamentals of CT and object oriented programming. This type of environment is proving to be ideal for CT paradigm development and at the same time allows students to develop their own creative ideas and collaborate with other members of a team (Zhang, Ordónez de Pablos & Zhu, 2012).

1.1.Computational Thinking

The history of Computational Thinking, as it is the case in many fundamental developments in science, reflects the convergence of multiple ideas, often from different areas of study (cognitive sciences, linguistics, psychology, computer science), that after being developed in isolation, found a synergetic effect when applied to the area of education, and in particular the processes involving generative languages for the creation of novel methods as well as complex systems.

Some of the pioneer thinkers in this field include Papert, Wing and Wolfram. One of the earliest references to CT is contained in (Papert, 1996) where Papert describes the value of applying human cognitive primitives to object oriented problems, by noticing the relationships between the components of a complex system. Other similar references can be found in (Vee, 2013; Wolfram, 2016) where there are direct references to the fundamental ideas of dividing a complex task into a set of simpler tasks.

We can find some specific descriptions regarding the core elements of CT in the work that ISTE[1] and CSTA[2], in collaboration with industry and K-12 education, have developed in order to address the needs of the educators that will eventually integrate this discipline in the classrooms. Some of the characteristics (Sykora, 2014) include: 1) Formulating problems in a way that enables us to use a computer and other tools to help solve them, 2) Logically organizing and analyzing data, 3) Representing data through abstractions such as models and simulations, 4) Automating solutions through algorithmic thinking (a series of ordered steps), 5) Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources, 6) Generalizing and transferring this problem solving process to a wide variety of problems.

The field of CT is still in its infancy; even if the overall goals and core principles are defined, two important tasks should be the object of research in the near future. One goal is to discover the uncharted world of cognitive processes that the human mind can sustain, provided the required tools. Cognitive science has dramatically altered the model for which the educational model of the past was designed. We know now (Pinker, 1995; Pinker, 1999; Kahneman, 2003; Kahneman, 2011) better than before what the limits of the mind are. The task ahead is to discover the computational resources, primitives, of the mind, in order to harvest their power. The second task is to develop a set of experiences that will allow students acquire these tools, engage in these cognitive processes, and produce rich and innovative results. The educational system, K-12 and beyond, is based on curriculum with pervasive use of descriptive languages and Type-A problems (Olabe, Basogain, Olabe, Maíz & Castaño, 2014). The new curriculum will be designed for the use of generative languages (object oriented languages) and Type-B problems. This task will require the collaborative effort of all fields participating in CT.

One educational system where some of these practical steps of integrating CT in the classroom is taking place is that of England in the United Kingdom. In England the national curriculum in CT addresses the objectives to educate their students to be able to: "1) understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation; 2) analyze problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems; 3) evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems; and 4) be responsible, competent, confident and creative users of information and communication technology," (Department for Education England, 2013).

In the United States the prevalent path for pre-university students to participate in CT activities and curriculum is the Advanced Placement Computer Science courses. These courses are offered in high schools, and upon graduation students receive college credit for introductory computer science courses. Some recent developments (Berkeley, 2017; Harvard, 2017) are expanding the enrollment of CT courses to university students of other fields. In coordination with these projects, high schools are offering courses in CT, such as ECE130, detailed in this paper, with the goal to provide dual-enrollment and university credit to students intending to study in fields such as the humanities, the arts, social sciences or business.

1.2. Blended Learning

The Virtual Learning Environments (VLEs) offer benefits to teachers and students. Teachers encourage collaboration and communication in the classroom, and they can customize and differentiate progress in the classroom. Students also benefit from the VLEs by learning to work with their peers on projects, and developing collaborative skills and problem solving.

[1] The International Society for Technology in Education (ISTE)

[2] The Computer Science Teachers Association (CSTA)

The VLEs also present some drawbacks including, among others, the intrinsic difficulty of the use of technology, the lack of technical and pedagogical support for the teachers, and the requirement of a formal decision and economic investment on the part of the educational institution. There are also some misconceptions regarding VLEs when they are considered as an alternative to traditional classes, or that they are exclusively a modality of online education (Paddick, 2014).

VLEs have been deployed at all levels of education with different stages of implementation. In higher education, the implementation is extensive, while in secondary education it has been limited to providing the technological infrastructure, and in primary education, with very few exceptions, it is almost non-existing (Johannesen, 2013).

This environment is beginning to change as a result of several developments: a) public and private schools are discovering the educational value of this new type of learning that includes multimedia, evaluation, monitoring, and collaboration; b) from a technical point of view, there is an evolution into VLEs that are easier to use and are adapted to the needs of students, and are easier to install and maintain locally or in the cloud; and c) the MOOC (Massive Open Online Courses) phenomenon (Breslow, Pritchard, DeBoer, Stump, Ho & Seaton, 2013; Vila, Andrés & Guerrero, 2014) and its variations (COOC, NOOC, SPOC) have revolutionized the way we use technology in distance and blended education (Benfield, Roberts & Francis 2006; Bruff, Fisher, McEwen & Smith, 2013).

The integration of CT in classrooms around the world will require the collaboration of efforts from diverse fields of academia, industry and government. It will also require the development of appropriate curricular materials and the training of teachers that will participate with their students in this major transformation of the educational system that was practically unchanged for generations. In this paper we describe some of the developments implemented in pilot projects, in primary and secondary education, as well as in pre-university education, that address important challenges in this transformation. In particular, we propose the use of a VLE such as Moodle as the technological platform to deploy innovative pedagogical strategies.

Moodle is open source and its functionalities are driven by the global community. It has been adopted by a significant segment of school systems around the world, and it presents continuous innovations and updates. It is powerful and secure, and provides developers and users an environment where it is possible to create effective online experiences for learning and teaching where collaboration is fomented and privacy is secured. Moodle is available in over 100 languages and has been tested and adopted by institutions of all sizes, economic backgrounds and geographic locations from around the world. It also supports the socially responsible collaboration between rich and poor communities via the Voluntary Service Overseas that stimulates global training opportunities.

The courses described in this paper have a common structure with the following core elements: 1) Video: a set of 4-6 video tutorials (3-5 minutes each) in which the concepts of the session are introduced, 2) Practice: Scratch or Alice project templates to allow the student the exploration of the project presented during the video tutorial. If necessary, the video tutorial would be revisited to achieve a complete understanding of the session, 3) Auto Test: Self-evaluation (5 minutes) by the student to determine the degree of acquired knowledge (it can be repeated as many times as necessary), 4) P2P (Peer-to-Peer) Task performed by the student to solve a problem by creating a Scratch or Alice project. The projects are evaluated by fellow students using a common rubric, 5) Test: Evaluative test (5 minutes) which measures the degree of knowledge acquired by the student (2 attempts), and 6) Explore and Discover: Scratch or Alice projects where students expand their knowledge discovering and exploring new ways to use Scratch blocks or Alice tools.

The methodology of the course is based on a learner centered paradigm. This implies that the tasks of the student include: a) following a series of short video lessons, b) taking interactive quizzes, c) assessing and being assessed through testing and Peer-to-Peer (P2P) and d) participating in online forums with classmates and teachers (Glance, Forsey & Riley, 2013).

These courses emphasize the experience of student learning. They highlight three important aspects of the design of the course: a) Multimedia content and Activity Scratch or Alice; b) Auto-test; and c) Tasks-P2P. The "concepts" and core "processes" of computational thinking that have developed in the course (College Board, 2016) are evaluated through the Test and P2P activities.

The design of the courses presented in this paper is based on a blended structure: the objective is to benefit from the qualities of the local teacher in the classroom (deep knowledge of each student, their learning styles and preferences, the optimal class dynamic, etc.) and to benefit from the services provided by the VLE (design and delivery of curriculum with multimedia, individual monitoring of the progress of each student, assessment and peer to peer evaluations, portfolio management, etc.).

Blended learning, with its substantial resources, is specially designed for differentiated instruction. To implement this type of selective instruction is necessary the creation of custom-designed instruction for each student: the first step is to assess the actual progress of each student, and then the optimal set of activities and assessments are programmed. It would be impossible for a teacher, given the limitations of time and effort, to attend to the individual needs and interests, abilities and learning styles of each student.

This ability to dynamically create several online options of classroom training for each student increases their ability to learn. It also improves the student satisfaction and participation in the class.

The use of blended learning, with its ability to continuously monitor the progress of each individual student, allows the pacing of new materials and assignments. And in cases where for health reasons the student is absent, blended systems are a friendly ally that helps keep track of past accomplishments and future tasks, providing guidance and planning where earlier options included only repeating the course.

The self-pacing features that blended systems offer have the effect of increasing the completion rate of students as compared with primarily e-learning environments, often affected by student lack of participation. Because the students in blended systems notice the immediate feedback of their progress they come to view their learning as a new continuous process and not a set of separate learning and disconnected events, which in turn increases the commitment and motivation of students, and their learning in the classroom.

2. PC-01 Course

2.1. Content

Our research team has focused its teaching and research activities of the last decade in the area of technology and e-learning education. The team was able to analyze, experiment with, and implement systematic studies of courses offered in these platforms (Olabe, Basogain & Olabe, 2016).

The team has designed and implemented the course 'PC-01: Introduction to Computational Thinking' with the following characteristics: a) immediate implementation in the school; b) simple access to the contents and tools by the teacher and students; c) basic introduction to concepts and processes in Computational Thinking; and d) efficient and sustainable use of educational technology.

The course has been developed in the learning platform "Egelapi" (https://egelapi.ehu.eus/), which is a VLE system based on Moodle (2.5.4). The course uses the Scratch 2.0 software as the programming language environment (online editor and Scratch 2 Offline editor).

The course includes the study of the following elements of Computational Thinking: 1) Computational thinking and expression (how to read and write in a formal language in order to solve problems). 2) Abstraction (how to communicate complex ideas simply, and logically break down problems). 3) Integration of multimedia content (text, images, sound, data, graphics). 4) Development of objects and functional blocks (objects, programs). 5) Interactive programs (events and event management). 6) Fundamental programming concepts (decisions, loops, variables, functions, sequential and parallel execution).

The course is organized into 10 sessions, each lasting two hours. The sessions are named according to the families of Scratch blocks being studied: Movement, Appearance, Sound, Pencil, Event, Control, Sensor, Operators, Data and More Blocks. Table 1 lists the collection of units implemented in the course PC-01.

Table 1. PC-01 Curriculum: Unit-level Content

Unit	Title
Unit 1	Movement
Unit 2	Appearance
Unit 3	Sound
Unit 4	Pencil
Unit 5	Event
Unit 6	Control
Unit 7	Sensor
Unit 8	Operators
Unit 9	Data
Unit 10	More Blocks

In addition, a Session 0 (it is called Initial Session) was created to familiarize the student with the learning platform Moodle and the programming language Scratch.

2.2. Assesments

The course PC-01 is based on a continuous student assessment and feedback and the creation of a student portfolio of projects and core concepts mastered. It includes a total of 10 tests, based on multiple choice questions of concepts, which are presented in text format, and programming paradigms and structures that contain programming scripts the student must analyze and identify.

The construction of the student portfolio is implemented with the cumulative collection of projects created as part of the P2P projects. These projects document the abilities acquired by the students, including the programming paradigms and the corresponding core ideas of CT. Table 2 lists the collection of 10 P2P projects implemented in the course PC-01.

Table 2. PC-01: List of P2P Projects

Project	CT Core Ideas
P2P 1 Design of a project to draw three triangles on the stage.	Sequence of instructions and repetition control.
P2P 2 Design of a project where the protagonist narrates a short personal story.	Use of costumes and text messages associated to a sprite.
P2P 3 Design of a project with 4 protagonists playing different sounds.	Use of multiple sprites (objects) and play sound procedures.
P2P 4 Design of a project where the protagonist draws on the stage images of creative art with lines and curves of multiple colors.	Controlling the pen resource of the sprite including the color and position.
P2P 5 Design of a project to allow the user to draw on the stage with a pen of multiple colors.	Use of Events and Event handlers with simple control loops.
P2P 6 Design of a project to implement a clock with a moving hand associated with multiple sound and graphic effects.	Control structures and introduction to multithread programming.
P2P 7 Design of a project with sprites ball, beetle, crab and beach ball interacting with each other using sensors of the environment.	Sensors that report information on the environment (keys pressed, asked question on the stage stored values, etc.).
P2P 8 Design of a project where a penguin exhibits mathematical talents.	Use of numerical and text operators, arithmetic, random, concatenation.
P2P 9 Design of a project to control the speed and color of a spacecraft by using variables.	Simple data manipulation (setting and changing variable content.)
P2P 10 Design of a project for the creation of new, user-defined programming blocks. The new blocks will allow the complex drawing of geometric patterns with variable parameters.	User designed blocks More Block: Custom blocks to abstract the functionality of scripts and make programming a modular task.

Figures 1 and 2 illustrate multiple choice questions presented to students during self-assessment and test sessions, including text-based conceptual questions and graphical script programming questions.

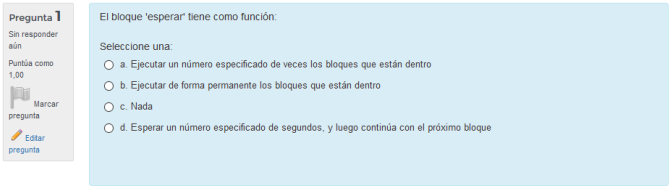

Fig. 1. Text-based conceptual question in Scratch.

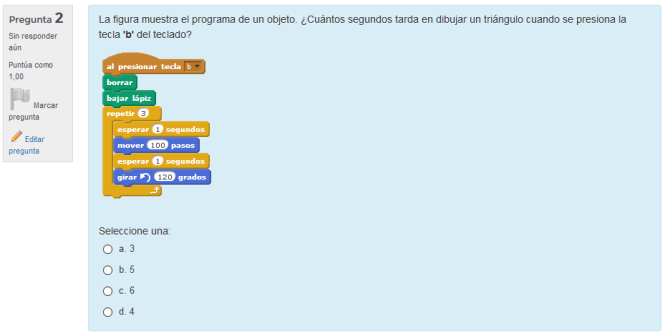

Fig. 2. Graphical Scratch script programming question.

The following figures illustrate solutions implemented by students in two P2P projects. Figure 3 illustrates the combination of a continuous script that controls the position of the drawing pen (controlled by the user via the mouse) and the parallel scripts, triggered by the keyboard, that allow the change of color and shape of the marks. Multiple event handlers provide these features in a parallel programming paradigm.

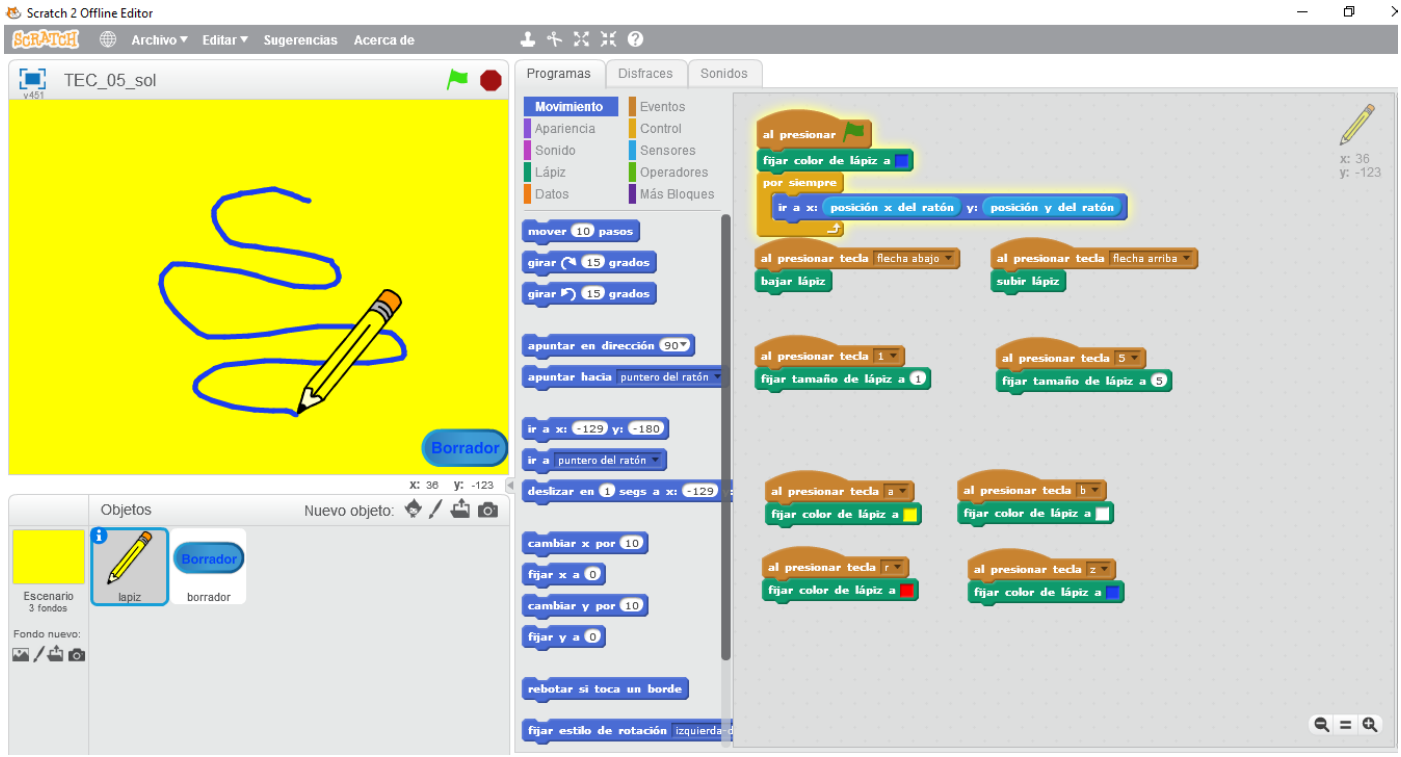

Fig. 3. Solution of P2P 5.

Figure 4 illustrates the use of multithreading programming to allow the user to control the velocity and appearance of a spacecraft by using variables. This project illustrates the powerful paradigm of decomposition of a complex task into simpler goals, each implemented with a custom script.

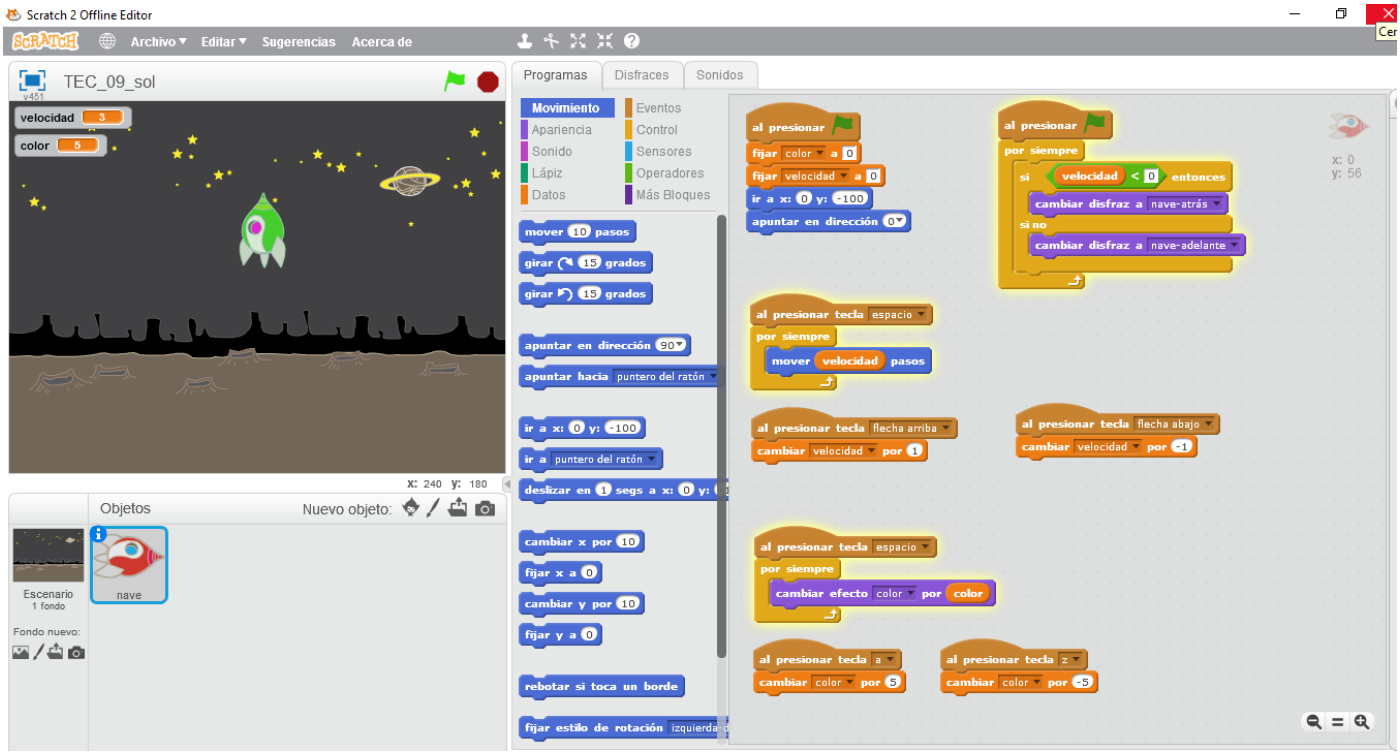

Fig. 4. Solution of P2P 9.

Students participating in the course PC-01 have successfully achieved the academic goals set in the project, as illustrated by their performance in the tests and the creation of their individual projects. In addition, students express a sentiment of satisfaction by their participation in activities they consider motivating and where they can express their different creative interests.

3. Course ECE130

3.1. Content

In addition to the course PC-01 (and other related courses, which are designed for students aged 10-15) our team also focused in the needs and challenges of the introduction of CT in the classroom of pre-university age students.

For this project we benefited from two especial characteristics that are preeminently present in the educational system of the United States. One is the ability of students in high school to earn college credit while in high school by taking Advance Placement courses (College Board, 2017) or by enrolling in dual-enrollment courses that offer credit both in the high school at the university level. The second is the increasing demand on computer literacy courses required in universities as part of the GER's (general requirement courses) of all students.

The University of one of the authors of this paper (CBU, 2017) has participated for many years with several high schools in dual-enrollment programs. In 2009 the course ECE130 (Introduction to Computational Thinking and Programming) was designed to introduce junior and senior students of high schools to the fundamental ideas of Computational Thinking while studying the core foundations of programming.

The course was designed as an introduction to object oriented programming using graphics in the creation of 3D movies, games and interactive applications. Alice was selected as the programming environment given the special features that it offered to the novice programmer. Very importantly, Alice eliminates all syntactical errors by using a menu driven interface that allows students to securely implement Java code while completely eliminating the demanding learning curve that most Java programming courses require. In addition, Alice is integrated into a 3D world where all objects represent familiar entities in everyday life: people, animals, vehicles, plants, houses, etc. This feature allows students to immediately assimilate object-oriented principles by association with state values, procedures and functions of real life objects, and their immersion in movies, games or interactive applications that imitate life systems and events.

The course is structured in twelve units that are listed on Table 3. Each unit includes a set of core ideas and principles that are illustrated via simple 3D projects that incrementally increase in complexity. The units include a set of video tutorials that students study and later master by replicating themselves in Alice the new learned ideas in the form of 3D projects. The process of studying the video tutorials and practicing in Alice is iterative and culminates when the student understands the process, and masters the implementation of the new ideas.

Table 3. ECE130 curriculum. Unit-level Content

Unit	Title
Unit 1	Introduction to object-oriented programming & software development
Unit 2	Program Design and Implementation
Unit 3	Algorithmic implementation
Unit 4	Abstraction and Optimization
Unit 5	Built-in Functions and Expressions
Unit 6	Classes and Objects, Messages and Methods, and Parameters
Unit 7	Interactive Programs, Events and Event Handling
Unit 8	Selection Structures
Unit 9	Boolean Functions
Unit 10	Repetition and Loop Control
Unit 11	Repetition and Recursion
Unit 12	Lists and List Processing

3.2. Assesments

After this initial mastery phase, the student needs to complete an open ended project in which the new ideas are integrated. These projects always take the form of a movie, a game or an interactive application. A comprehensive rubric is provided as a guide for the student. This rubric will later be used to Peer assess the work of other students.

The weekly assessment is implemented with three separate tools: self-assessment, test, and Peer to Peer Assessment. The self-assessment is a tool that allows students to review a comprehensive checklist of core ideas, new vocabulary, formal programming rules and regulations that need to be mastered before proceeding further. The tests are independent assessments that document the progress of students and certify the achievement required for a successful completion of the course.

The Peer to Peer assessments address several student skills in the process of acquiring the status of a Computational Thinker. One developed quality is the ability of read code, to interpret knowledge represented in the form of programming code, and to assess its complexity and quality. By assessing the work of others, and comparing the assessments with the rubric assessments, the student develops expertise in the skill of reading, interpreting and evaluating the quality of written code.

A second skill developed by the Peer to Peer assessments transforms the traditional passive role of the student in which his or her work is evaluated by the teacher into an active role in which the student is given the privilege and the responsibility of evaluating the work of others. This important responsibility conveys the implicit message that educating the student means making the student an expert: an expert that can create quality work, and an expert that can appreciate and evaluate the work of others. Rarely in traditional education is a student expected to be able to reliably assess the work of others, and rarely in traditional education is a student trained, week by week, in the demanding and important task of been an expert with the skill to reliably assess the work of others.

Table 4 includes a list of the ten P2P projects of the course ECE130.

Table 4. ECE130: List of P2P Projects.

Project	CT Core Ideas
P2P 1 Design of a project with 4 Scenes and 2 Sub-scenes. Leader object defines action to be implemented.	Top-down design and bottom up implementation. Decomposition.
P2P 2 Design of a project using Class level Methods. Use of Markers to identify physical locations.	Objects contain behaviors that can be expanded with new Class Methods.
P2P 3 Design of a project using user-defined functions. Functions will obtain information directly form the environment and pass it to the main program.	Objects can determine useful information from the environment through the use of user-defined functions.
P2P 4 Design of a project with Class Methods and Functions that use Parameters.	Class methods and functions provide adaptive behavior through the use of parameters.
P2P 5 Design of a project with the use of control structures for individual and collective segments of code.	Control structures add versatility to the code of individual objects and collections of objects.
P2P 6 Design of a project with nested groups of control structures for individual objects and collections of objects.	Complex topologies of nested control structures add functionalities to languages.
P2P 7 Design of a project with data structures including one and two dimensional arrays of objects.	Data Structures allow the manipulation of families of objects efficiently and dynamically.
P2P 8 Design of Project using Random strategies for the control of Arrays.	The use of Random functions allows the resolution of problems through trial and error strategies.
P2P 9 Design of an interactive project by interfacing with objects through events and event handlers.	Events and event handlers allow the implementation of dynamic systems.
P2P 10 Design of a comprehensive project including all the core elements of the course.	Complex dynamic systems, using arrays of objects and interfacing with the external world can be implemented in modular form.

Figures 5 and 6 illustrate examples of multiple choice questions that students answer in their weekly self-assessment assignments or in the more formal test sessions. These questions include a combination of text based conceptual questions or the analysis and evaluation of graphical programming scripts in Alice.

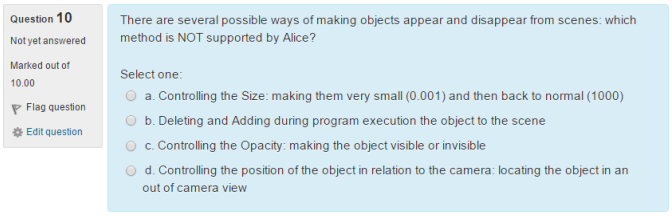

Fig. 5. Text-based conceptual question in Alice.

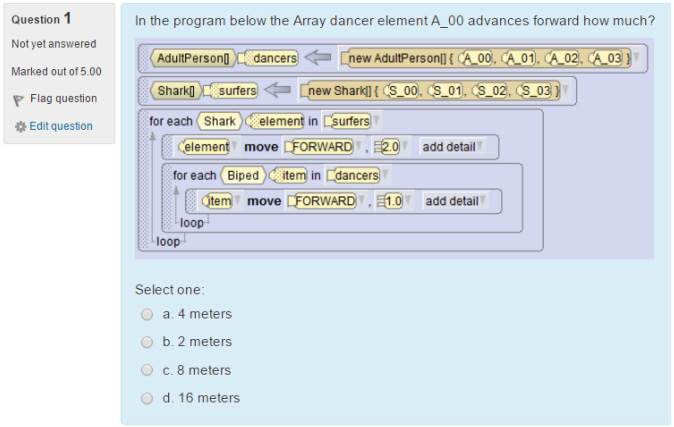

Fig. 6. Graphical Alice script programming question.

The following figures illustrate the programming environment in Alice with some student programming examples. Figure 7 shows a partial view of a program that is structured into six scene methods, showing and interactive dialog between a fortune reader and a group of students, where random complete sentences are created.

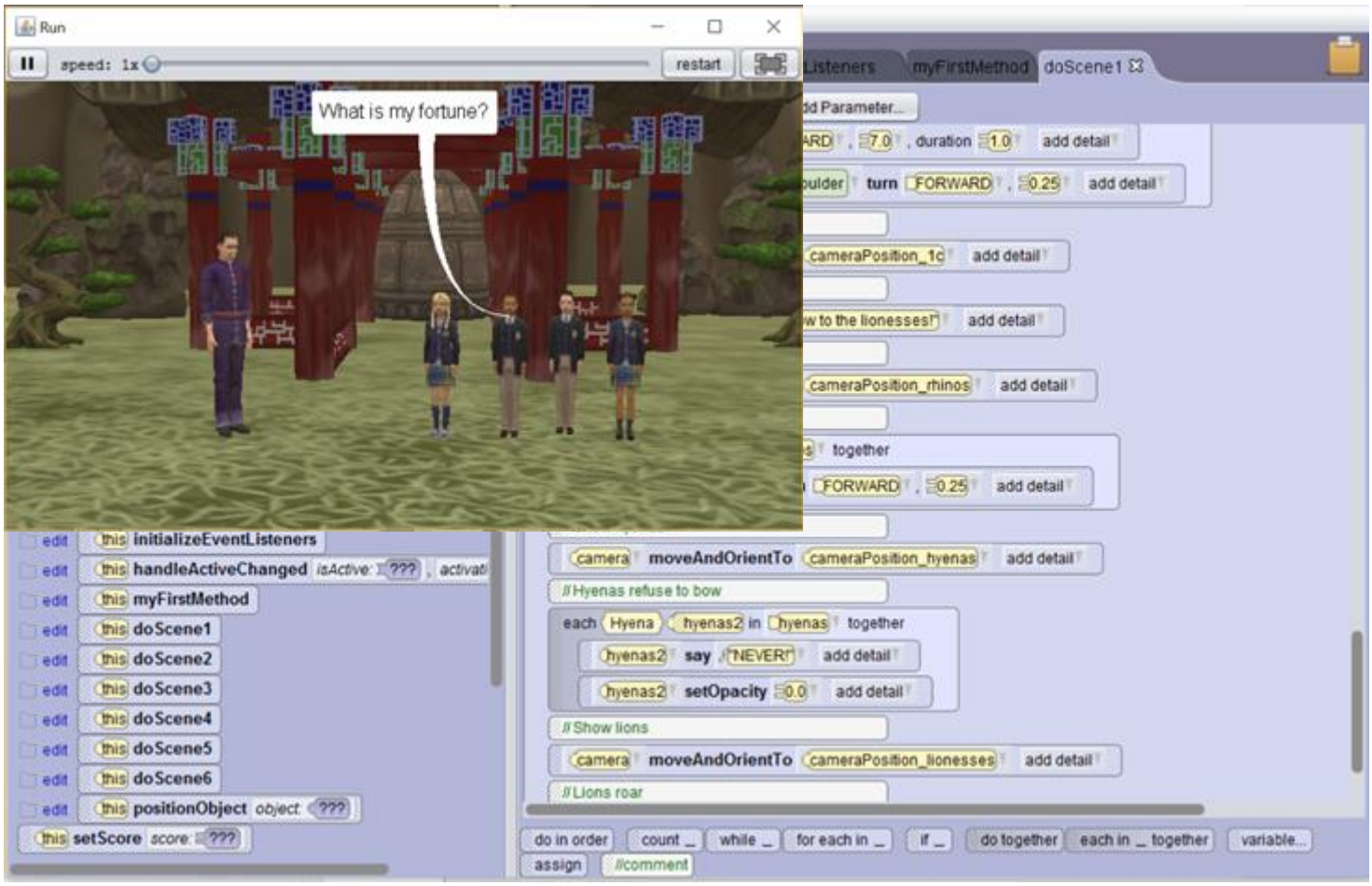

Fig. 7. Random Sentence Generation Scene in Alice.

Figure 8 shows a set of three arrays of animals programmed to perform actions individually within each array, in unison for each array, and in coordination with other arrays.

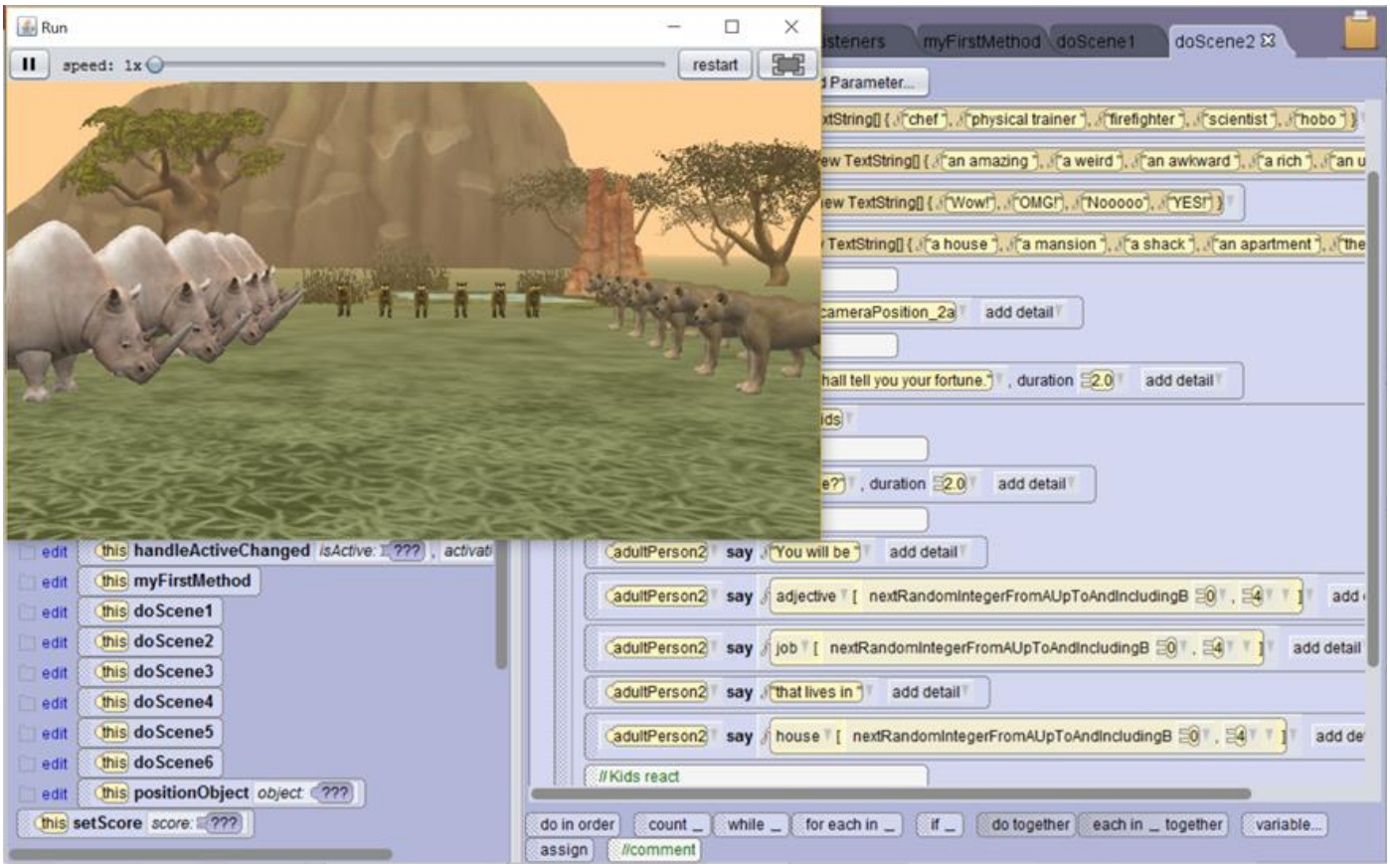

Fig. 8. Programming Object Arrays in Alice.

The programming environment of Alice provides students recently introduced to the world of programming a selected set of tools that, given their simplicity, are easy to learn and manipulate, and at the same time allow the rapid prototyping of fairly complex programs with a multiplicity of interacting objects.

4. Discussion

The integration of Computational Thinking curriculum and activities in the educational system will require the transformation in some fundamental ways of the relationship between schools and society, the role of the teacher in the classroom, and the students' experience during their period of education. We now discuss some of the events involved in this transformation and the impact on institutions, teachers and students.

4.1. Impact on Educational Institutions

The final structure of the courses described includes a software set of resources that are easy to install in a VLE and that offer a comprehensive set of educational services in the area of Computational Thinking for primary and secondary students.

The course curriculum provides educational institutions with a specific and avant-garde proposal of "concepts" and "processes" in the area of computational thinking that can be easily introduced in primary and secondary education and high school. The courses are taught in a blended format in the computer laboratory and with the support of the school VLE system.

Currently, the course PC-01 is hosted in a virtual classroom EDUCANDO online (http://aula.educando.edu.do), which is the portal under the supervision of the Department of Computer Education of the Ministry of Education of the Dominican Republic (MINERD).

These types of academic initiatives provide institutions a platform to experiment with offerings of extracurricular activities where students and parents can evaluate the costs and benefits of participating in new academic fields, such as Computational Thinking, which are not readily available in traditional school offerings. In addition, educational institutions participating with these types of courses can incrementally incorporate new technologies in their academic structures and project to society an image of a state of the art institution.

The VLE platform serves as the central academic repository of all students' resources, activities and projects allowing the administrators to evaluate in an integral form all schools associated with the project (Sancho Gil & Padilla Petry, 2016). At the same time the VLE environment provides

means for sharing resources among the schools, backup installations, deployment of institutional badges, and other centralized services.

These services are made possible by features present in Moodle, and other environments, that automatically generate reports on students' activity, participation, and learning goals achievement in wide set of learning analytics. Some of these features are part of the standard distribution of Moodle, and other more specialized services are provided by third-party plugins which are available in the plugins directory (Moodleplugins, 2017).

The availability of Learning Analytics is fundamental in the process of improving the learning outcomes of an educational institution. These analytics directly provide feedback and information not only to students and teachers but to administrators and decision makers as well.

An example of higher level learning analytics is the course overview. This tool allows to rank multiple courses by activity, participation and other criteria. This type of comparative analysis is of relevance for medium and large size institutions in determining the efficiency of different policies and strategies, and it would be almost impossible to implement with traditional mechanisms.

4.2. Impact of Teachers and Teaching Loads

One of the fundamental changes in education with the assistance of VLE environments is the creation of the collaborative mode of education, where students receive their knowledge from both the local teacher, present in the classroom, and the multimedia based knowledge embedded in the VLE. For example, to offer Computational Thinking courses in an institution, it is not necessary for the local, classroom teacher, to be an expert in the field. What it is important is that the teacher be knowledgeable in the use of the VLE. The content of the course, assessments, projects and other academic tasks are provided by what we could call the "Guest Teacher" through the VLE environment. The local teacher implements tasks of classroom leader, motivator, and moment by moment director of activities.

The teachers in these courses have a set of resources and services that facilitate the delivery and monitoring of an introductory course in Computational Thinking. The teachers have access to collections of video tutorials, self-tests, tests and P2P projects covering the scope of the course curriculum. Before starting the course the teachers receive training on methodology and course content.

To fully assist the local teacher, a comprehensive guide has been created describing the structure and operation of the main parts of the course. This documentation includes a description of the topics studied in the course, the pedagogical resources available, how the course is organized into modular sessions, and how these sessions need to be presented to the students. In the areas of assessment, both individual assessment and Peer to Peer evaluation, the documentation provides guidelines for teacher and students on the optimal use of these resources. Because Peer to Peer evaluation is in general a new tool for most institutions, special emphasis is dedicated in the description of the three phases of evaluation, the criteria for grading the submitted work, and how the final grades are obtained. Finally, a comprehensive index includes the available resources (video tutorials, practice projects, self-tests, tests, and the solutions to the Peer to Peer projects.)

Teachers also find a number of integrated analytics learning tools in the VLE (Moodle) that allow them to implement collective and individual student progress assessments of "concepts" and "process" in computational thinking (Singh, 2015).

In these courses we have installed a new block called Progress Bar that provides a graphic representation of the activities completed by the students in reference to all the activities in the course. This tool provides an ideal feedback to students, allowing them to see with a color coded system the activities as well as the resources completed and those yet to complete.

The structure of the Progress Bar is tailored by the teacher who has the ability to include from all preexisting resources those appropriate to monitor. Then, the analysis can be made under different criteria: completed, approaching deadlines, etc. This allows teachers to have on a single page a comprehensive summary of the complete progress of all students in the class. This in turn allows to evaluate the general progress of the course, the identification of isolated cases, as well as the early detection of at-risk students. The often selected activities included in the Progress Bar are: Self-assessment, Assessment, and Peer to Peer Evaluation.

The following figures (Figures 9, 10 and 11) illustrate some of the features VLE environments offer in order to provide more timely feedback to students and teachers, to recognize and reward student achievement, and facilitate mechanical and time consuming tasks in class management. Figure 9 illustrates the access to Progress Bar for both students and teachers.

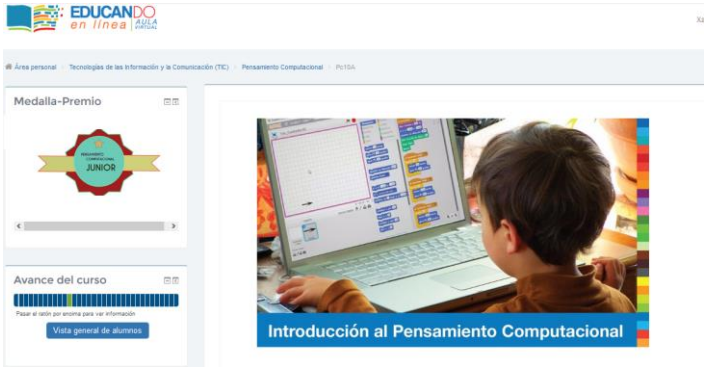

Fig. 9. Access to the Progress Bar

Similarly, the teacher has access to the records of total and partial scores of students. The management tools allow the grouping of tasks into different collections in order to obtain relevant information on the completion and progress of the course (see Figure 10).

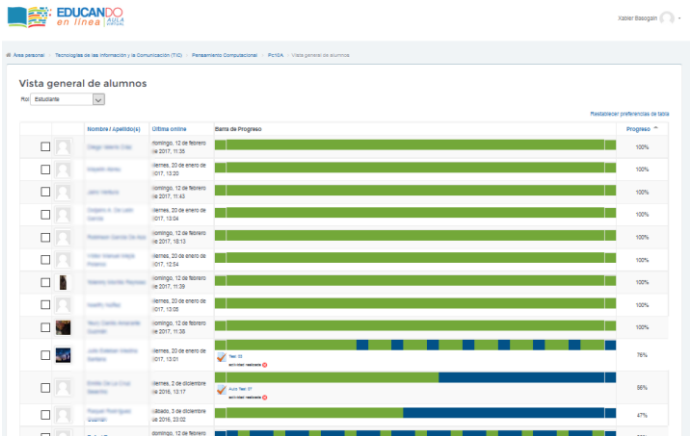

Fig. 10. Progess Bar: teacher's tool

An important time saving tool for teacher is the automatic calculation of grades. Because these courses are designed with a continuous evaluation of the progress of the students, as well as the creation of their portfolio by means of their programming projects, each student generates a vast number of individual assessment values. The overall grades illustrated in Figure 11 are obtained automatically from all the individual assessments according to a rubric that weighs the different aspects of the course. For example, the criteria for each session may include a final grade that is obtained as a combination of 50% of the test, 40% of the P2P assessed value, and 10% P2P grading task.

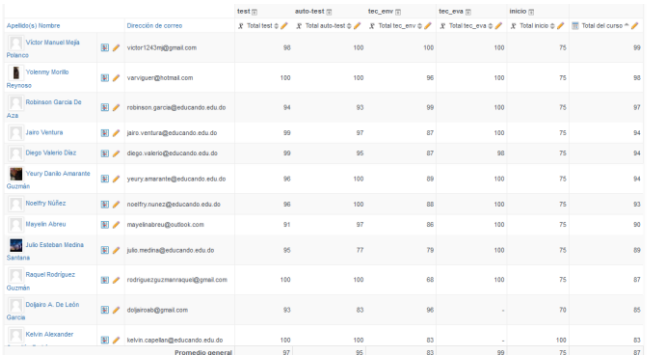

Fig. 11. Grades of students

4.3. Impact on Students

The courses described in this paper are characterized by the fact that the students construct their own knowledge following the pedagogical theory of constructionism. This pedagogical theory proposed by Seymour Papert (Papert, 1991) proposes that students build their knowledge through the construction of an 'artifact' that motivates them. For the construction of an 'artifact' the courses use the programming environments Scratch or Alice and their digital editor. Each course unit contains two sections, called Practice and P2P.

In the Practice section students experiment in their programming environment with the conceptual contents developed by the teacher in the video tutorials; in the P2P section, the students must design and build a project to solve a situation or a proposed problem. Then, they build a solution to the problem using the contents and concepts learned in the unit.

One aim of the courses is for students to learn the principles of a computer programming language. In these courses the students use Scratch or Alice as a tool that allows them to communicate and create ideas to solve problems (Disessa, 2000).

The implementation of a Scratch or Alice project that solves a problem requires the development of higher cognitive functions of the mind. These functions include categorization, decision making, abstraction, insight, problem solving, planning and execution.

The second part of the P2P tasks, the assessment of programming projects of 3-5 fellow students, according to a specific rubric, is implemented anonymously, and it promotes as well the development of higher cognitive functions. The students are required to play the role of evaluators, and in turn deepen their knowledge in order to evaluate Scratch or Alice projects created by other students.

The VLE environment of the course allows the student not only to play the role of evaluator but also to collaborate with peers giving and receiving feedback, indicating errors, solutions and possible improvements.

In addition to being exposed to other Scratch or Alice projects, these activities open for the student the opportunity to learn and be inspired by new solutions for future problems (Lu & Law, 2012).

The experience of the students in these hybrid courses also allows them to become familiar with the tools and methodologies of the VLE environment. As students of a VLE system they have experienced different aspects of this form of learning: multimedia format content, collaboration mechanisms, self-assessment and evaluation mechanisms, progress in their knowledge and grades. The courses offer a badge (see Figure 9, in top of the Progress Bar), a prize or medal to recognize accomplishments and provide students with a reward that acknowledges their achievement in Computational Thinking (Seliskar, 2014).

In the immediate future these students will see their own knowledge progress through the use of tools and services of educational online platforms offering massive MOOC courses. Lifelong learning will require students the use of VLE environments similar to the environment experienced by the student in these courses (Attwell & Hughes, 2010).

4.4. Participants & Results

During the last several years, the courses described in this article have been implemented in classrooms of high schools in the United States (continuous evaluation during the academic years from 2009 until 2016) and in secondary schools in the Dominican Republic (Study-1: April-June 2016, and Study-2: December-2016/March-2017).

Study-1: Dominican Republic: The PC-01 project was implemented during a six month period. The first part included the design, implementation and fine tuning of the Learning Management Platform "Egelapi". The course was later replicated on the platform EDUCANDO online of the Ministry of Education (MINEDU). In this platform,

classrooms for ten schools in the area of Santo Domingo were created.

The second part of the project involved the delivery of the course in 4 schools. Previously, the teachers of the schools were trained to facilitate the course. The course is taught by the teacher to her students in person in the classroom with the support of the learning platform. The 4 schools began the course after the Easter holidays. The schools have taught the course according to their own schedules and time availability. The duration of the course varied, and lasted between 6 and 10 weeks. Each classroom assigned 10 students, with an average of 8 active students.

The outcomes of the course were assessed using the portfolio of each student and the number of successful tests. These parameters indicate a high degree of success. Other course results were obtained from personal interviews with students and teachers, and they indicate a high degree of satisfaction on the part of the course participants (some testimonials: 'easy', 'funny', 'entertaining', 'learn more', 'create') (Basogain, Olabe, Olabe, Ramírez, Del Rosario & Garcia, 2016).

The second study in the Dominican Republic included 21 schools distributed across the country in the areas of Santo Domingo, Pedro de Macorís, San Juan de Maguan, San Francisco de Macorís y Puerto Plata. Each participating school included 1, 2 or three groups of students, with students of primary and secondary education with ages ranging from 10 to 15. Table 5 lists the participant's schools.

Table 5. Participating Schools in the PC-01 project in RD.

School	Name
1	Coronel Rafael Tomas Fernández Domínguez
2	Damián David Ortiz A , B
3	Dr. José Francisco Peña Gómez
4	Básica Cañada Grande A,B,C
5	Carmen García García
6	María Altagracia Paula
7	Nery Cueto De Delma
8	Primaria Francia Margarita Ayala
9	Primaria Juan Pablo Duarte
10	Primaria Padre Sillas
11	Sor Leonor Gibb
12	General Antonio Duverge
13	Liceo JE José Joaquín Pérez
14	Manuel María Castillo
15	Padre Brea
16	Profesora Jacoba Carpio
17	Sergia María Mateo A, B
18	Vicente Aquilino Santos
19	Lic. Vespertino Gregorio Luperón
20	Básico Padre Eulalio A. Arias – Pax
21	Fidel Ferrer

With the conclusion of one study and the beginning of a new one, the lessons learned are incorporated to introduce improvements in those areas identified as priority. In particular, special effort has been dedicated to the implementation of the Peer to Peer project and assessments, and the activation of the different phases: submission, assessment, grading, and closing. These tasks were supported with informative video tutorials and training session via Skype.

An important class management tool offered by VLE's is the ability to process and analyze the numerical data of the students' level of learning and achievement. In particular, in the course PC-01 we have analyzed: 1) the number of self-assessment and test attempts by the students; and 2) the corresponding grades obtained in those attempts. It is relevant to note that students may attempt the self-assessment as many times as they wish, without direct impact in their grades. The tests, however, can only be attempted twice, with the best grade being recorded. To illustrate the results obtained in this study, we include the following two figures (Figures 12 and 13) which show the results of two particular Moodle classrooms (named PC10 and PC12) located in the cities of Las Matas de Farfán and San Pedro Macorís.

Figure 12 shows the total number of attempts in self-assessments and tests in these classrooms. One can observe the general descending tendency throughout the ten weekly sessions of the course. This seems to indicate that students initially choose to practice for longer periods of time when the subject is still new and unknown. And as the course advances, students experience an increased confidence on their skills and knowledge, and therefore feel less need to complete additional attempts. There is a visible difference between both classrooms, but the difference is more quantitative than qualitative.

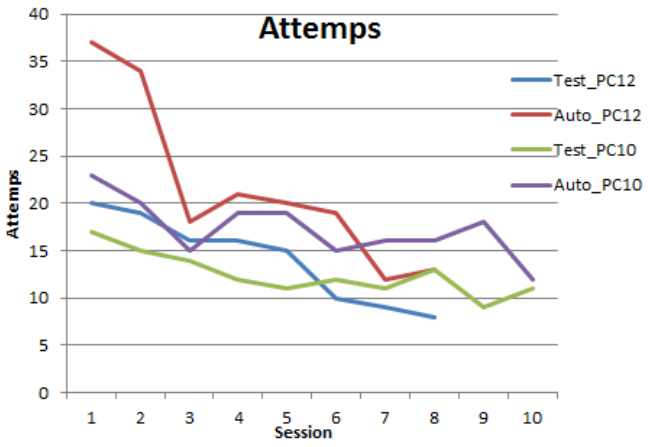

Fig. 12. Attemps of Self-Assessment and Test of PC-01 Course.

Figure 13 shows the grades obtained both in the self-assessments and in the tests. One can observe a general tendency of obtaining better grades on the tests than on the self-assessments through all 10 weekly sessions of the course. This seems to indicate that the students gradually reinforced their knowledge on core fundamentals as they accumulated experience and time on these fundamentals with the passage of time, attempted assessments and effort dedicated to project designs.

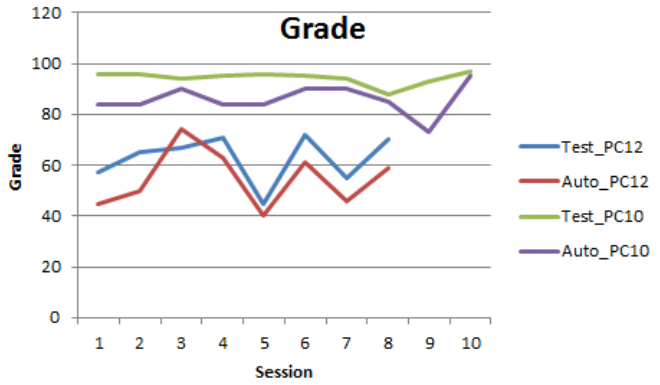

Fig. 13. Grades in Self-Assessment and Test of PC-01 Course.

These graphics seem to indicate other effects as well: a) There seems to be a different work attitude in the two classrooms, with one consistently working more than the other; b) also, some sessions, for example session-5, can be identified as more problematic given a substantial drop in the average grade.

The numerical values of these graphics are included on Table 6 (values on a 100 point scale).

Table 6. Numerical values of Grades

	Autotest		Test	
	Grade	**Attemps**	**Grade**	**Attemps**
PC12A	54,75	21,75	62,75	14,13
PC10A	85,9	17,3	94,4	12,5

The course ECE130 is traditionally offered to several high schools in the Memphis, TN area, and includes Christian Brothers HS, White Station HS. St. Georges HS, Overton HS, Saint Benedict HS, Central HS and Briarcrest HS. Each high school participates with one or two sections of students in their Junior or Senior year, with ages 16 to 18.

Because the teaching schedule of ECE130 is fairly intense (one hour of lecture per day, five days a week) with the corresponding daily homework, special effort is dedicated to guarantee that assignments are completed before the deadline. High school students in US are expected to travel with frequency to participate in sport and cultural activities sponsored by various associations, and therefore the online format of these courses is ideally designed for flexible scheduling of class work.

Given the open environment in which these courses are offered, special effort is dedicated to guarantee that a large set of randomized tests and assessments are available to prevent the possible data sharing among students.

Throughout the years the course ECE130 has experienced an increase in the complexity of the Peer to Peer projects as students adapt ever more rapidly to new technologies and evaluation methods using computers.

4.5. Conclusions and future directions

In this paper we present two courses on Computational Thinking for primary, secondary education and high school. These courses introduce the "concepts" and "processes" of computation in a hybrid format of classroom and online education.

The area of knowledge of Computational Thinking is experiencing a significant expansion in the private and public education sectors both in the developed world and in the developing world. Students trained in Computational Thinking are significantly better prepared for the daily tasks and the professional work that awaits them in their immediate future.

The area of educational technology is also experiencing a breakthrough in new services and resources for the training of students. The MOOC phenomenon is revolutionizing the world of education and training. Many students, who for various reasons could not access education, today can benefit from quality training and success.

In addition, educational computing offers us every day new products and educational tools that facilitate the task of teaching those materials that until now could not be included in the school curriculum.

These courses represent real examples of synergy between computer science and educational technology. The courses use the educational potential of VLEs systems to train primary, secondary and high school students in the new curriculum subject of computational thinking. The courses exploit the analytical tools of learning and they allow the teacher to successfully perform the tasks of teaching, monitoring and grading the students. On their part, the students have a learning experience in a constructionist environment of collaboration and are able to monitor and track their own progress.

The course PC-01 has been scheduled to be taught in ten public schools in Santo Domingo and its surrounding areas during the third term of the academic year 2015-16. The results of the course have been evaluated with the goal of improving it before making its deployment in public schools across the Dominican Republic.

This experience of designing and implementing an introductory course of computational thinking in primary and secondary education is a pioneer step by a university research group that closely works with a technical team of the department of educational computing at the Ministry of Education of the Dominican Republic.

The work and effort dedicated by both teams is part of the tasks of cooperation between the scientific community and the educational community for improvement of the education of our young students.

The course PC-01 is made available to any educational initiative that aims to introduce Computational Thinking in primary and secondary schools in the around the world. Educational institutions interested in implementing the PC-01 course may contact via email the authors of the article.

High school students of the US that complete college-credit courses, such as ECE130, during their education, already begin their university careers with fewer credits to complete. In addition, an increasing number of universities include in their General Requirement Courses for graduation a

course in Introduction to Programming or Computational Thinking.

At the same time, institutions that offer dual enrollment courses, such as CBU, create permanent partnerships with High Schools, thereby establishing academic pipelines that supply steady source of students enrolling in their university programs. Probably the major forces behind dual-enrollment partnerships involve the prestige of HS offering college courses and the supply of students enrolling in universities.

PC-01 in Colombia: A new project of the course PC-01 has been created between the University of the Basque Country and the Corporation for the National Academic Network for Advance Technology (RENATA) in Colombia via an agreement of collaboration with the title "Introduction of Computational Thinking in the Schools of Bogota and Colombia". In addition, the Ministry of Information Technologies and Communications (MINTIC) of Colombia has joined the collaboration agreement, providing resources for the implementation of the project in 10 schools distributed across the country (see Table 7).

Table 7. Colombian Schools participating in the PC-01 Project.

School	Name
1	Juan Hurtado, Belén de Umbria – Risaralda
2	Pedro Uribe Mejía, Santa Rosa - Risaralda
3	Cadena Las Playas, Apartadó - Antioquía
4	24 de Mayo, Cerete - Córdoba
5	Antonio Nariño, Moniquira - Boyacá
6	Niño de Jesús de Praga, Girón - Santander
7	San Rafael, Soledad-Atlántico
8	Augusto Medina, Ibagué - Tolima
9	Nuestra Señora de Guadalupe, Dosquebradas - Risalda
10	INEM, Pereira - Risaralda

This project in Colombia is being developed during the second semester of the academic year 2016-17, and it is supported by a Moodle-based platform (Renata, 2017), and it is illustrated in Figure 14.

Fig. 14. Platform PC-01 in Colombia

The course ECE130 also is part of an expansion program in order to provide services to a larger group of High Schools in Memphis and surrounded areas, as well as High Schools traditionally linked to CBU in Nashville, Chicago, St. Louis and New Orleans. This expansion project is part of a university wide project of K-12 STEM outreach.

References

Alice, (2017). An Educational Software that teaches students computer programming in a 3D environment. Retrieved from: http://www.alice.org

Attwell, G., & Hughes, J. (2010). Pedagogic approaches to using technology for learning: *Literature review.*

Barbour, M., Brown, R., Waters, L. H., Hoey, R., Hunt, J. L., Kennedy, K., & Trimm, T. (2011). Online and Blended Learning: A Survey of Policy and Practice from K-12 Schools around the World. *International Association for K-12 Online Learning*.

Basogain, X., Olabe, M. A., Olabe, J. C., Ramírez, R., Del Rosario, M. and Garcia, J. (2016). PC-01: Introduction to Computational Thinking. Educational Technology in Primary and Secondary Education. 2016 *International Symposium on Computers in Education (SIIE).* pp. 1-5. http://dx.doi.org/10.1109/SIIE.2016.7751816

Benfield, G., Roberts, G., & Francis, R. (2006). The undergraduate experience of blended e-learning: a review of UK literature and practice. *London: Higher Education Academy.*

Berkeley, (2017). The Beauty and Joy of Computing. Retrieved from: http://bjc.berkeley.edu/

Breslow, L., Pritchard, D. E., DeBoer, J., Stump, G. S., Ho, A. D., & Seaton, D. T. (2013). Studying learning in the worldwide classroom: Research into edX's first MOOC. *Research & Practice in Assessment*, 8.

Bruff, D. O., Fisher, D. H., McEwen, K. E., & Smith, B. E. (2013). Wrapping a MOOC: Student perceptions of an experiment in blended learning. *Journal of Online Learning and Teaching*, 9(2), 187.

CBU, (2017). Christian Brothers University. (2017). Retrieved from: https://www.cbu.edu/

Code.org, (2012). Anybody can learn. Retrieved from: http://code.org

College Board, (2016). AP Computer Science Principles. Course and Exam Description. Retrieved from: https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf

College Board, (2017). AP Courses. Retrieved from: https://apstudent.collegeboard.org/apcourse

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, 58, 240–249.

Department for Education England, (2013). "National curriculum in England: computing programmes of study - key stages 1 and 2". Ref: DFE-00171-2013. Retrieved from: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study

Disessa, A. (2000). Changing minds: Computers, learning, and literacy. *Cambridge: MIT Press*.

Ertmer, P.A., Ottenbreit-Leftwich, A.T., Sadik, O., Sendurur, E., & Sendurur, P. (2012). Teacher beliefs and technology integration practices: A critical relationship. *Computers & Education*, 59, 423-435.

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M.U., Bontá, P., & Resnick, M.(2013). Designing scratchjr: Support for early childhood learning through computer programming. *In Proceedings of the 12th International Conference on Interaction Design and Children ACM*. pp. 1-10.

Glance, D. G., Forsey, M. & Riley, M. (2013). The pedagogical foundations of massive open online courses. *First Monday*, 18 (5). Retrieved from: http://firstmonday.org/ojs/index.php/fm/article/view/4350/3673. doi: 10.5210/fm.v18i5.4350

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2), 199-237. doi: 10.1080/08993408.2015.1033142

Harvard, (2017). CS50: Introduction to Computer Science. Retrieved from: https://cs50.harvard.edu/

Hubwieser. P., Armoni, M., Giannakos, M. N., & Mittermeir,R. T. (2014). Perspectives and visions of computer science education in primary and

secondary (k-12) schools. *Transactions on Computing Education*, 14(2):7:1{7:9

Johannesen, M. (2013). The role of virtual learning environments in a primary school context: An analysis of inscription of assessment practices. *British Journal of Educational Technology*, 44: 302–313. doi: 10.1111/j.1467-8535.2012.01296.x

Kahneman, D. (2003). Maps of Bounded Rationality: Psychology for Behavioral Economics. *The American Economic Review*, vol. 93 no. 5, pp. 1449-1475. doi: 10.1257/000282803322655392

Kahneman, D. (2011). Thinking, fast and slow. *Macmillan*.

Kalelioğlu, F. (2014). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, Volume 52,Pages 200-210, ISSN 0747-5632, http://dx.doi.org/10.1016/j.chb.2015.05.047.

Lu, J., & Law, N. W. Y. (2012). Understanding collaborative learning behavior from Moodle log data. *Interactive Learning Environments*, 20(5), 451-466.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L. & Settle, A. (2014). Computational Thinking in K-9 Education. In ITiCSE '14 *Proceedings of the 2014 conference on Innovation & technology in computer science education*. (pp. 1-29). doi: 10.1145/2713609.2713610

Moodleplugins, (2017). Moodle plugins. Retrieved from: https://moodle.org/plugins/

Olabe, J.C., Basogain, X., Olabe, M.A, Maíz, I. & Castaño, C. (2014). Solving Math and Science Problems in the Real World with a Computational Mind. *Journal of New Approaches in Educational Research*, vol.3 no. 2, pp. 75-82. doi: 10.7821/naer.3.2.75-82

Olabe, J. C., Basogain, X. & Olabe, M. A. (2016). Developing New Educational Frontiers through Breakthroughs in Cognitive Computation and New Dimensions in Pedagogical Technology. *International Journal of Social Science and Humanity*, vol. 6, no. 11, pp. 813-820. doi:10.18178/ijssh.2016.V6.755

Paddick, R. (2014). As easy as VLE. Education Technology. Retrieved from: http://edtechnology.co.uk/Article/as_easy_as_vle

Papert, S. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), Constructionism. 1-11. *Norwood, NJ: Ablex*

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1):95{123,1996.

Pinker, S. (1995). The language instinct: The new science of language and mind. *Penguin UK*, vol. 7529.

Pinker, S. (1999). How the mind works. *Annals of the New York Academy of Sciences*, 882.1, pp.119-127.

Renata, (2017). Red Nacional Académica de Tecnología Avanzada, Aula Virtual - Curso PC-01. Retrieved from: http://168.227.244.48/moodle/

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60-67.

Sáez López, J.M., González, M.R. & Cano, E.V. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "scratch" in five schools. *Computers & Education*, doi: 10.1016/j.compedu.2016.03.003.

Sancho Gil, J., & Padilla Petry, P. (2016). Promoting digital competence in secondary education: are schools there? Insights from a case study. *Journal Of New Approaches In Educational Research*, 5(1), 57-63. doi:10.7821/naer.2016.1.157

Seliskar, H. V. (2014). Using Badges in the Classroom to Motivate Learning. Faculty Focus. *Magna Publications*. Retrieved from: http://www.facultyfocus.com/articles/teaching-with-technology-articles/using-badges-classroom-motivate-learning/

Singh, J. (2015). Learning Analytics tools available in Moodle. Retrieved from: http://www.moodleworld.com/learning-analytics-tools-available-in-moodle-moodleresearch-moodleworld/

Sykora, C. (2014). Computational thinking. Operational Definition of Computational Thinking for K–12 Education. *ISTE website*. https://www.iste.org/explore/articleDetail?articleid=152

Vee, A. (2013). Understanding computer programming as a literacy. *Literacy in Composition Studies*, 1(2):42{64,2013.

Vila, R. R., Andrés, S. M., & Guerrero, C. S. (2014). Evaluación de la calidad pedagógica de los MOOC. *Profesorado: Revista de curriculum y formación del profesorado*, 18(1), 27-41.

Wolfram, S. (2016). How to Teach Computational Thinking. *Blog Stephen Wolfram.* Retrieved from: http://blog.stephenwolfram.com/2016/09/how-to-teach-computational-thinking/

Zhang, J. X., Liu, L., Ordónez de Pablos, P., & She, J. (2014). The auxiliary role of information technology in teaching: Enhancing programming course using Alice. *International Journal of Engineering Education*, 30(3), 560–565.

Zhang, J. X., Ordónez de Pablos, P., & Zhu, H. (2012). The impact of second life on team learning outcomes from the perspective of IT capabilities. *International Journal of Engineering Education*, 28(6), 1388–1392.

Acknowledgment

The authors would like to express their gratitude to the members of Department of Computer Education of the Ministry of Education of the Dominican Republic (MINERD) for their collaboration, and to the members of RENATA and Ministry of Communications and Technologies (MINTIC) of Colombia.
This work was supported in part by the Research Development Grants of the University Basque System (2016-18), Department of Education, Universities and Research – Basque Government, Spain