Recommender Systems and Scratch: An integrated approach for enhancing computer programming learning

Jesennia Cárdenas-Cobo, Amilkar Puris, Pavel Novoa-Hernández, José A. Galindo and David Benavides

***Abstract*—Learning computer programming is a challenging process. Among the current approaches for overcoming this challenge, visual programming languages (VPLs), like Scratch, have shown very promising results for beginners. Interestingly, some higher-education institutions have started to use VPLs to introduce basic programming concepts, mainly in CS1 courses. However, an important issue regarding Scratchs usage in higher education environments is that students may feel unmotivated being confronted by programming exercises that do not fulfill their individual expectations. To try and overcome this barrier, we propose CARAMBA, a Scratch extension including an exercise recommender system. Based on features like *taste* and *complexity*, CARAMBA is able to personalize student learning with Scratch by suitably suggesting exercises for students. An in-depth evaluation was conducted about the effects of our proposal on both the learning of basic concepts of CS1 and the overall performance of students. We adopted an equivalent pretest-posttest design with 88 college students at an Ecuadorian university. Results confirm that recommending exercises in Scratch had a positive effect on students programming learning abilities in terms of pass rates. In totality, the pass rate achieved by our proposal was over 52%, which is 8% higher than the rate achieved during a previous experience using only Scratch (without recommendation) and 21% higher than the historical results of traditional teaching (without Scratch). Furthermore, we analyzed the degree of exploitation of CARAMBA by students to portray two facts: students actually used CARAMBA and there was a significant, positive correlation between the utilization of CARAMBA and the scores obtained by the students.**

***Index Terms*—Scratch, recommender systems, visual programming languages, programming learning**

I. Introduction

NOWADAYS, learning computer programming remains a challenging task. It generally comprises of hard skills related to problem solving (e.g., modeling, abstraction, logical thinking, and coding), which are difficult to teach or learn. To cope with such difficulties, several approaches have been suggested in the past, which include collaborative work [1], simulation tools [2], games [3], pair programming [4], and role games [5].

In this context, visual programming languages (VPLs) have proven to be important tools for beginners with little or no experience in programming. This is the case among most children and high-school students. By mapping conventional programming concepts into visual metaphors, VPLs have significantly impacted the learning of computer programming [6]. Popular VPLs like *Alice* [7], *RAPTOR* [8] and *Etoys* [9] have been used at different educational stages. *Scratch* is perhaps the most widespread VPL today [10], [11]. Although it was initially conceived for children, few higher-education institutions have adopted it for introductory programming courses [12]–[14].

This was the case with the *Universidad Estatal de Milagro* (UNEMI) in Ecuador, where Scratch was employed as a complementary tool to consolidate introductory courses on programming (CS1). The decision to use Scratch in UNEMI was based on the following two reasons:

- the reported success of Scratch in similar educational contexts; and
- the students low performance in CS1 since the beginning of the Computer Science Engineering program in 2003 (as the pass rate in UNEMI has historically been under 50%).

Using Scratch over the last two semesters, the pass rate has not only increased, but it has also stabilized. Notwithstanding the positive experience, some issues regarding its utilization were observed. We found that certain students felt unmotivated with the assignments (exercises) suggested by the lecturer during the course. For instance, when high-performance students were faced with simple exercises or vice versa. A similar experience was reported by [15].

To overcome this issue and exploit the benefits of Scratch, in this study we have presented a novel system called *CARAMBA*. It extends Scratch by including a recommender system for suggesting exercises (problem statements) to the student. Based on a collaborative-filtering approach, this intelligently suggests exercises to students based on their interests and level of assessment. During proposal, we perceived high levels of student satisfaction regarding the proposed system. Through simple questionnaires, the students believed that Scratch and a recommender system for exercises would help improve their overall performance in CS1. Despite such a positive reception, it was considered that a more serious analysis was needed to gain better insights on the true impact of the proposal. Thus, we have statistically validated the effects of the recommending exercises with Scratch on both the learning of programming concepts and the overall performance of the students. An

J. Cárdenas-Cobo is with State University of Milagro, Guayas, Ecuardor. email:jcardenasc@unemi.edu.ec

A. Puris and P. Novoa-Hernández are with the Technical State University of Quevedo, Quevedo, Los Ríos Ecuador. They are also researchers at the State University of Milagro.

Jos A. Galindo and D. Benavides is with the Department of Computer Languages and Systems Universidad de Sevilla, E.T.S.I. Informática, Av. Reina Mercedes s/n, 41012, Seville, Spain email: {jagalindo,benavides}@us.es.

equivalent group design was adopted to analyze the three different learning strategies:

1) traditional;
2) Scratch-only; and
3) Scratch plus recommending of exercises (CARAMBA).

It is worth mentioning that this study complements our preliminaries results [16] in two major directions: 1) the development of an improved (and more mature) system, and 2) the prosecution of a deeper analysis about the effects of this system on student learning.

From the results obtained we concluded that recommending exercises in Scratch has a significant and positive impact on the learning of CS1 concepts in college students. Moreover, it improves the overall performance of the students. We especially showed that CARAMBA surpasses the traditional approach by 21% and the Scratch-only approach by 8%.

This paper is structured as follows: Section 2 reviews related studies, while Section 3 presents the main features of CARAMBA as an integrated approach, including technical aspects of the implemented recommender system. Section 4 is devoted to the validation of CARAMBA, while Section 5 outlines the discussion of the results and a conclusion of the research.

II. Related Work

In this Section, we go through related works about both Scratch and Recommender Systems at the university level. It is important to note that most literature related to Scratch is about experiences with elementary and high school students. This is to be expected, since Scratch was conceived for children and teenagers. However, as we will show in the next section, from the very beginning, few researchers have been interested in how to exploit it for enhancing programming learning in CS1.

A. Scratch in college environments

Perhaps, the first study using Scratch in higher education was [10]. Here, the authors included Scratch in the Harvard Summer School: *The Great Ideas in Computer Science* (a summer school version of a course at Harvard College). The goal was to improve first-time programmers experiences in CS1 by devoting one week to explaining general CS concepts using Scratch. Afterwards, students would transition to Java for the remainder of the course. From conducted surveys, the authors concluded that "Scratch excites students at a critical time (i.e. their first foray into computer science)". Scratch also "familiarizes the inexperienced among them with fundamentals of programming without the distraction of syntax".

In a similar study, [12] reported an experience at an Uruguayan university. This study aimed to analyze the use of Scratch in a CS1 course, before introducing the Java programming language. To improve learning with Scratch, the author proposed specific exercises to be solved with the same. By conducting both a qualitative and quantitative analysis, the author concluded that "the use of Scratch in the very beginning of the CS1 course promoted a high level of motivation, thus a positive perception of learning programming". However, the author did not observe "any measurable improvement of the results obtained by students who used the tool compared with the normal course". The same holds true for the pass rates, where no significant differences existed. The author justified these outcomes by the fact that students of the Scratch course were affected by two (learning) "jumps": Scratch-to-Java, whilst the control group was only affected by one "jump": Java. In other words, it seems that the benefits acquired by starting with Scratch disappeared when students had to face Java in the middle of the course. This seems to suggest that a transitional education strategy from Scratch to Java (or any other conventional programming language) is required to overcome the "two-jump" syndrome.

The attitudes and opinions of future teachers about Programming and Information and Communications Technology (ICT) in Education were explored by [17] using Scratch in a CS1. Perhaps the most relevant aspect of this research is that it was restricted to women studying preschool education and educational design. The authors observed that Scratch was considered user-friendly by students and helped to increase their interest in exploiting ICT in education. Consequently, the stress and anxiety related to using ICT in educational practices tended to decrease.

In [18], the role of Scratch in teaching CS1 was discussed. The author specifically explored how Scratch could be used in business programming. Although the paper reported preliminary results, the author confirmed that Scratch "allows educators to reduce the cognitive load that students experience when first introduced to programming".

A broader research, as presented in [13], assessed the impact of a short intervention by Scratch in a CS1 course. The idea was to use Scratch to facilitate the introduction of a more complex programming language, in this case C++. The study involved 332 first-year undergraduate engineering students and the authors recorded both the quantitative and qualitative data for assessing the impact of Scratch. Analysing the collected data, the authors concluded that a two-week Scratch intervention, along with specific educational resources (e.g. labs and projects), would be beneficial for learning basic programming concepts. Thus, both the novices and advanced learners could be satisfied. However, it was additionally noted that such an intervention did not help novices obtain grades similar to those of advanced learners when faced with typical programming exam questions.

A similar study was conducted by [19] aiming to facilitating the learning of programming concepts by undergraduate Computer Engineering students. The study was developed at the State University of Feira de Santana and involved a particular learning approach "based on peer support, game development, a challenge-response strategy and Scratch". Through several surveys and the analysis of source code the authors found that students considered Scratch to be "user-friendly... with good working mechanics, especially the Lego-style blocks".

Two positive experiences were reported in [20] and [21] regarding introducing programming to pre-service teachers with Scratch. After attending theoretical lessons, students developed an educational app and wrote a report highlighting both the negative and positive impressions on their individual experiences. As a result, it was observed that students posi-

TABLE I
CARAMBA VS. OTHER CONTRIBUTIONS FROM LITERATURE.

	Recommender system	Evaluation	Tool extension	VPL
[9]	–	–	–	Etoys
[8]	–	–	–	Raptor
[7]	–	–	–	Alice
[10]	–	√	–	Scratch
[24]	–	√	√	Scratch
CARAMBA	√	√	√	Scratch

tively assessed important Scratch's features.

Recently, Martinez et al [22] presented a negative experience in which authors used the extension Dr. Scratch in the first weeks of a video-games course. Results showed that students were not additionally motivated using Scratch as students associated Scratch with pre-university level courses and not serious programming.

Few important conclusions can thus be drawn from the above experiences:

1) The common practice in college education when using Scratch is to perform a short intervention, as the first interface between a novice and complex-to-learn programming languages;
2) Additional educational strategies are required to perform an effective intervention with Scratch. In other words, using Scratch as it was originally conceived for children is not enough to cope with the expectations of college students regarding computer programming; and
3) No previous literature exists about extending Scratch technologically to enhance programming learning in colleges. Although there are similar experiences, like Dr. Scratch [23], [24], they are not employed in college scenarios.

Thus, to provide better insights about the features of available systems for enhancing computer programming through VPLs, in the Table I we compared the same including our proposal CARAMBA. The features for consideration were: 1) the presence of a recommendation system, 2) whether the research was evaluated or validated, 3) whether it extended the employed VPL, and 4) which VPL was used.

B. Recommender systems and learning environments

This Section is inspired by the related work of our previous work presented in [16]. An RS is a software tool that determines and suggests what a particular user would find useful [25]. It is also considered to be a part of the so-called information filtering system, which exploits user information for predicting ratings or preferences that the user would give to an actual item [26], [27]. Thus, the basic benefit of an RS is that it finds the most suitable set of items for a target user by maximizing its rating prediction.

According to [25] five types of RSs exist: content-based, knowledge-based, demographic, community-based, collaborative, and hybrid. Collaborative Filtering RSs (CFRSs) have probably been the most widely used [28], [29]. These were built on the assumption that one user might like items that other users with similar tastes had liked in the past. This assumption was adopted for the RS that was implemented in the present work. CFRSs involve two major approaches: user-user and item-item [29]. In general, both approaches rely on the Nearest Neighbors algorithm [28].

A major area where RSs have been broadly applied is e-learning environments, within the context of TEL [30], to improve students' autonomous learning. While in e-commerce domains RSs suggest products or services to clients, in e-learning environments RSs suggest educational resources (e.g., papers, books, or courses) to educational participants (e.g., students and/or teachers).

Related literature shows a wide variety of works proposing RSs for e-learning environments. Few important works in the field are briefly explained further. For an in-depth survey, the reader is to refer to [31] and, more recently, to [32].

The RS *CourseAgent* was conceived with the idea of enabling students to provide feedback in implicit and explicit ways [33]. This RS allowed students to directly evaluate courses with respect to (i) their relevance regarding career goals and (ii) the level of difficulty of the course. Both parameters provided implicit feedback when students planned or registered for a course. The basic and evident benefits of this RS for students were as follows: (i) it constituted a course management system that retained information about the classes taken, and (ii) it facilitated communication with student advisors.

The Virtual University of Tunis develops automatic recommendations in e-learning platforms [34]. These are composed of two modules: an offline module that pre-processes data to build learner and content models; and an online module that uses those models *on the fly* to recognize students' needs and goals to predict suitable recommendation lists.

It has been argued that traditional RSs are not suitable for supporting e-learning since they have not yet taken into account two important mechanisms: the learning processes and the analysis of social interaction [35]. To deal with these issues, the authors of the argument proposed a flexible approach involving a multidimensional recommendation model and a Markov Chain model. Results showed that more suitable recommendations could be obtained from such an approach. Similar research with a personalized approach was proposed, which relied on data mining and natural language process technologies to determine learner relationships based on the learning processes and learning activities [36].

To guide learners in personalized, inclusive e-learning scenarios, an important analysis was conducted on how RSs could be applied to e-learning systems [37]. Here, three technological requirements for developing semantic education RSs were provided. Other authors have additionally reported successful experiences using similar ideas. See for example [38]–[41].

A framework for the rapid prototyping of knowledge-based RSs was reported in [42]. This was used for recommending learning objects. From a software development perspective, the proposed framework was flexible enough to implement new approaches and included default implementations of al-

ternative strategies for each of its five stages. Two RSs were implemented to illustrate the benefits of this framework.

Related to the technological benefits of RS in education, there are subjective dimensions to this topic, which are also important for study. For instance, a psychological view of learning with personalized RSs was provided by [43]. Here, a very good fit between the main features of RSs (collective responsibility, collective intelligence, user control, guidance, and personalization) and the principles in learning sciences was demonstrated. However, the authors claimed that a "recommender systems should not be transferred from commercial to educational contexts on a one-to-one basis, but rather need adaptations in order to facilitate learning." In this context, few potential adaptations were grouped into system-centered adaptations (e.g., for enabling proper functioning in educational contexts) and social adaptations (e.g., for addressing typical information processing biases).

Similar to the previous work, in [44] a conceptual framework was proposed to explain how evolving recommender-driven online learning systems (ROLS) support students, course authors, course instructors, system administrators, and policy makers in development. Moreover, this framework involved two important perspectives in the constructivist paradigm of learning: *cognitive* and *situative*.

Additionally, an interesting approach to enhance RSs in collaborative learning environments has been presented, which consists of an influence diagram including observable variables for assessing the collaboration among users [45]. By applying machine learning techniques, the influence diagram was refined to increase its accuracy. The main outcome was the development of an automatic RS alongwith a pedagogical support system in the form of a decision tree, which provided visual explanation to the user.

Furthermore, a generic cloud-based architecture for a system that recommends learning elements according to the affective state of the learner was presented [46]. The authors also provided some use cases, explaining implementation. Undoubtedly, this is an interesting technological solution for exploiting cloud-based learning environments, which is a common feature in many education institutions.

An important survey on how to evaluate RSs was conducted in the context of TEL [47]. From an in-depth survey obtained from 235 works on the subject, it was concluded that there exists an important interest to design better strategies for evaluating RSs in TEL. Future trends and research opportunities were also highlighted in the study.

Summarizing the above review, three important conclusions can be drawn:

1) using RS in educational environments is a popular research topic with increasing associated studies;
2) most of the existing works are technology-based, where RSs are proposed for enhancing the learning process. Contrarily, just a few works employ a subjective perspective like [43] to analyze the role of RS in this context; and
3) as per our knowledge there are no RSs supporting programming learning with Scratch.

Conclusions 1 and 3 thus provide additional reasons for proposing the system explained next.

III. CARAMBA: An integrated approach

The proposed system aims to improve the current state of teaching Foundations of Computer Programming at Universidad Estatal de Milagro, Ecuador, with Scratch. Figure 1 helps demonstrate both the current and the proposed approaches for such a process.

In the first approach (Fig.1-a), the professor interacts with students exclusively by means of classroom lectures (formal learning). The students use Scratch in the traditional form, by creating or modifying projects (informal learning). Despite the benefits of this method, the professor often encounters difficulties in controlling the efficiency of the students interaction with Scratch (i.e., whether the students are properly traversing the knowledge levels in which the course is organized). In light of these facts, a gap between the informal and formal learning approaches is observed. Additionally, as observed over the course of one year, certain students have not been satisfied with the complexity of the exercises they were asked to solve with Scratch. For instance, the more experienced students were faced with Scratch projects that they deemed too easy.

A possible solution for these issues is a personalized set of exercises for students. (An exercise is defined in the context of this research as a problem statement that the student can solve in Scratch.) However, this demands for the individual characterization of students to assign to them the most suitable set of exercises, according to their knowledge level and expectations. This is a difficult task for the professor, mainly because there are too many students and exercises to assign. Moreover, since student learning is a dynamic process, both the student characterization and the suggested exercises are expected to evolve over time. Hence, this assignation process is to be repeated over and over.

An alternative solution is to increase the amount of time in laboratory practice with the professor being present. Thus, the professor could control the student's interaction with Scratch. However, this would put too much emphasis on formal learning, which contradicts our education goals. The challenge is thus how to improve the current approach with the least level of professor intervention possible. More specifically, it is important to find solutions to issues with the current approach while maintaining benefits of the employed learning approaches.

In keeping with this consideration, we propose CARAMBA, a Web Application composed by Scratch as a project editor along with an RS for exercises (see Figure 1-b)). Please notice that this proposal not only allows for the interactions in the current approach, but also personalizes the learning process of students using Scratch. By this, the professor is able to control students learning processes by creating exercises and including them in the system. Furthermore, students have the opportunity to assess the exercises in order to inform the system of their personal preferences. Using this information, the system suggests new exercises to the student, assuming that students with similar tastes and complexity perceptions about exercises are a good source for recommendations.

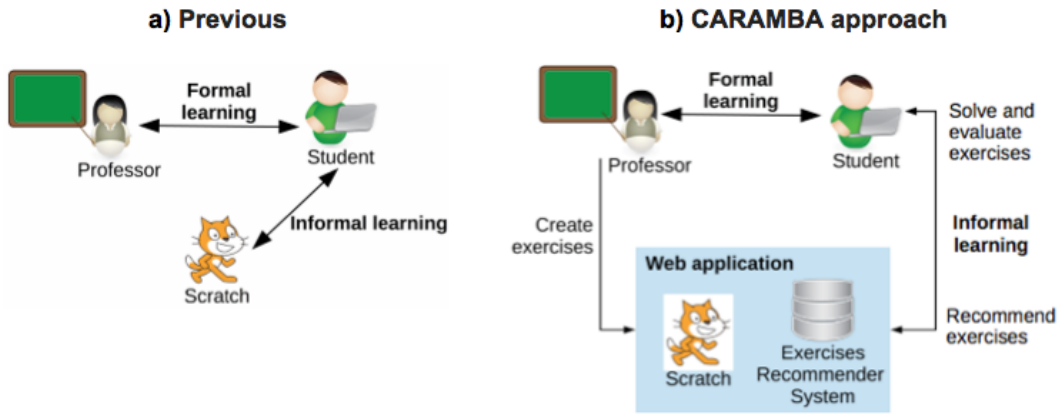

Fig. 1. Comparison between the Scratch-only approach and the CARAMBA approach.

From a pedagogical perspective, this proposal is a new mediator between professors and students. In the following section, the proposal is presented in detail, where the developed Web Application is described through its main modules and features and then the technical aspects of the implemented RS are explained.

The main workflow of the interaction between the student and the proposed system is depicted in Figure 2. After logging in, the system checks whether the student has previously seen an exercise. If they have, 10 exercises are recommended following a collaborative filtering approach. Otherwise, the system selects the easiest and most interesting exercises among all available exercises. The system additionally builds a list of 10 exercises that are to be presented to students.

The following steps are easy to understand by referring to the diagram. However, it is important to note that after submitting the exercise evaluation, the system applies a collaborative filtering to provide a new pool of recommended exercises to the student. Consequently, the student may face different exercises according to their individual experiences.

The student profile is thus built based on his/her assessments (Taste and Complexity) about the solved exercises. In the next subsection we provide additional technical descriptions about the RS for exercises. Following this, we briefly describe the main functionalities of CARAMBA.

A. Technical aspects of the recommender system

The RS included in our Web Application is based on a collaborative filtering (CF) approach, using both users and items [26], [27] and relying on the surprise library implementation artefacts (http://surpriselib.com/). In this context, the users are the students, while the items are the exercises to be solved in Scratch. The aim of this RS is to exploit the experience of existing students and suggest suitable exercises to a certain student. In the following sections, we refer to such a student as an *active student*.

To build the CF model, we assume that we have a set of n students $\{u_1, u_2, u_3, ..., u_n\}$ and a set of m exercises $\{e_1, e_2, e_3, ..., e_m\}$, with each student assessing an exercise according to the following two criteria:

1) *Taste* ($I \in \{1, 2, 3, 4, 5\}$). This measures how interesting the student found the exercise. A value close to 1 means that the exercise is not interesting at all for the student, while a value of 5 means the opposite.
2) *Complexity* ($C \in \{1, 2, 3, 4, 5\}$). This states the complexity level of the exercise from the student's point of view. For this variable, 1 means a lower complexity, 2 a medium complexity, and 3 a higher complexity.

In this manner, each student would have a record of viewed exercises along with their corresponding evaluations. By using this record, the goal is to determine which students have common exercises and an evaluation that's similar to other students. Once this first filter is applied, the next step is to recommend the exercises that are evaluated by other students but not yet viewed by the active student in question. Such a recommendation system involves the following processes:

1) Cold start: During the implementation of an RS in real environments, a common difficulty is how to recommend exercises when no user experience or data exists. This issue is known as *Cold start* [48]. In the case of our RS, we have two scenarios:

- *the system has no recorded user experience*. In this case, a random set of exercises is proposed to the active student from the pool of all available exercises; and
- *the active student has no experience recorded by the system*. This is the case for new users. Here, the system recommends exercises that are most frequently evaluated by the community with $I = 5$ and $C = 1$. This assures that new users start with the most popular and easiest exercises.

To reduce the lack of exercises in the early stages of the CARAMBA adoption we have manually created sets of exercises based on interactions from previous years before introducing the recommender system. We first obtained the sets of exercises used before using the recommender system within CARAMBA. Secondly, we asked students to evaluate each exercise and then introduced these exercises in the system to reduce the number of cold starts during the course. It is worth highlighting that the students who evaluated our initial

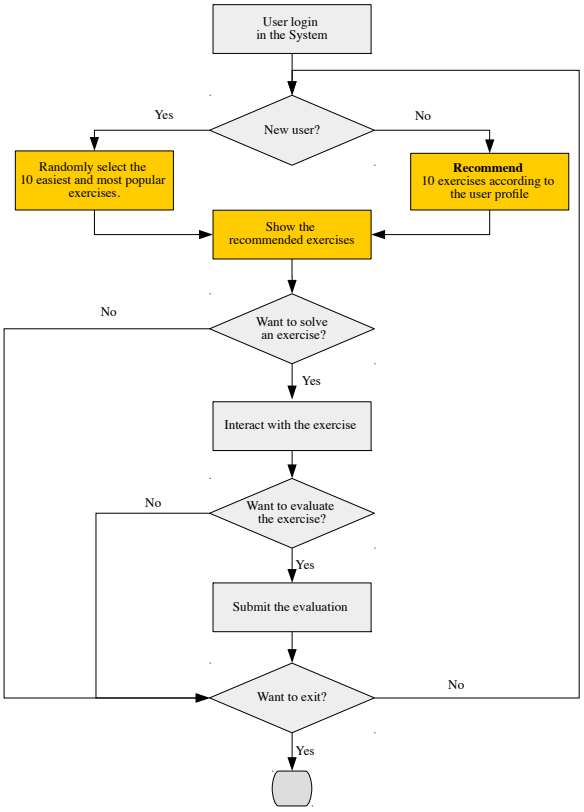

Fig. 2. Main flow of CARAMBA.

batch of exercises did not participate in the study presented in Section IV-B.

2) Neighborhood computation: A user-user collaborative filtering approach is adopted here. As mentioned above, the first step consists of finding the students who are the most similar with respect to the active user k. To this end, the Cosine Similarity function between two vectors [49] is considered. These vectors contain evaluations made by two students with common exercises, according to a certain criteria (e.g., taste or complexity). This measure is then applied to each evaluation criterion independently.

More formally, $S^{(t)}$ and $S^{(c)}$ are values of similarity computed for the evaluation of criteria Taste (t) and Complexity (c) respectively, and v_k and v_i are the evaluation vectors for the exercises that the active student u_k and the student u_i have in common. Thus, the corresponding similarities between students u_k and u_i are computed as:

$$S^{(t)}(u_k, u_i) = \frac{v_k^{(t)} \cdot v_i^{(t)}}{\left\|v_k^{(t)}\right\| \cdot \left\|v_i^{(t)}\right\|} \tag{1}$$

and

$$S^{(c)}(u_k, u_i) = \frac{v_k^{(c)} \cdot v_i^{(c)}}{\left\|v_k^{(c)}\right\| \cdot \left\|v_i^{(c)}\right\|} \tag{2}$$

$S^{(t)}$ and $S^{(c)}$ take values in the range $[0, 1]$. A value closer to 1 means a high similarity between the students, while a value closer to 0 means the opposite.

It is important to highlight that Equation 1 is not able to express the significance of the number of common exercises with respect to all previouslt recorded exercises. Since Equation 1 computes similarities using information from the exercises that both the students have in common, it does not take into account their corresponding records of exercises. We argue that this information is crucial to sutably compute similarities between two given students. In this regard, Equation 3 was employed:

$$S^{(p)}(u_k, u_i) = \frac{|H_k \cap H_i|}{|H_k|} \tag{3}$$

where, H_k and H_i are the sets of exercises seen by students u_k and u_i respectively. It is easy to note that Equation 3 quantifies the significance of the number of exercises that the active user k has in common with user i by computing the percentage of common exercises regarding the record of the active user. This expression is also defined in the range $[0, 1]$ and possible values have the same meaning as Equation 1.

Thus, we have three sources for computing the similarity for every pair of students: $S^{(i)}$, $S^{(v)}$, and $S^{(p)}$. The next question is how to aggregate them to obtain a single value portraying overall similarity. Several alternatives exist to deal with this. For instance, an average or weighted sum of the three similarity values could be used. Another approach is to

multiply them:

$$S(u_k, u_i) = S^{(t)}(u_k, u_i) \cdot S^{(c)}(u_k, u_i) \cdot S^{(p)}(u_k, u_i) \quad (4)$$

Notice that since $S^{(t)}$, $S^{(c)}$ and $S^{(p)}$ take values in $[0, 1]$, then S will also take values in this range.

3) Building the list of recommended exercises: This is a key stage in the usage of recommender systems. CARAMBA relies on the Weighted Sum of Others Ratings [50] to calculate a value (considering taste and complexity) for each exercise not evaluated by u_k . Equation (5) shows this task, where $(\bar{r}_{u_k})$ is the averaged value of u_k for each problem solved and $(\bar{r}_{u_i})$ is the averaged value for each similar user u_i.

$$r_{u_k,e} = \bar{r}_{u_k} + \frac{\sum_{u \in \Upsilon}(r_{u_i,e} - \bar{r}_{u_i})S(u_k, u_i)}{\sum_{u \in \Upsilon} S(u_k, u_i)} \quad (5)$$

Where:

- $r_{u_k,e}$: estimated value for the exercise e by the user u_k,
- $S(u_k, u_i)$: similarity result 4
- $r_{u_i,e}$: estimated value of the user u_i for the exercise e and,
- Υ: set of similar users to u_k $(S_{(u_k,u_i)} > 0)$.

Later, the recommendation algorithm selects the top 10 exercises with more value and generates the recommendation list for u_k.

Table II summarizes the main steps of the recommendation process, which corresponds to *Recommend* in Figure 2-b).

B. Main functionalities of CARAMBA

CARAMBA is a web application developed to manage the process of recommendation, solution, and assessment of exercises. This web application has an easy-to-use graphic user interface (GUI) and the current version was developed using Python and the PostgreSql Server as a database system.

Among the main features of CARAMBA, one can mention that:

- It presents a user authentication area , where you can enter previously registered students and teachers.
- In the student interface, an informational panel (see left panel Figure 3) where students can manage their profiles and obtain a summary of the exercises carried out and the assessments of those exercises
- The central area ((Figure 3) was designed to show the description of the selected exercise and its evaluation parameters.
- The area to the right (Figure 3) presents a list of recommended exercises, a list of all exercises, and a chronometer that is activated with the start button and measures the solution time for the selected exercise.
- The Scratch tool is activated in a pop-up window (Figure 4) through the "Start" button (see central area in Figure 4).
- In the teacher profile teachers can add new exercises and obtain statistical information about the exercises and the students.

Finally, is important to mention that the RS was included in CARAMBA using *Surprise*[1], which is a Python Scikit (short for SciPy Toolkits) for building and evaluating recommender systems.

[1]http://surprise.readthedocs.io/en/stable/index.html

IV. Evaluation

The studies carried out in the Universidad Estatal de Milagro assess the effects of the computational tools on learning. They were organized in two stages. At the first stage, we developed a qualitative study based on satisfaction levels in using the tool; this was applied in the 2015-2016 period. At the second stage (2016-2017), we conducted a quantitative analysis of the effect on the development of a skill set and its influence on pass rates. These studies are presented in detail below.

A. Motivation and Satisfaction study

In this section, we will show the results of a study *Real-Life Testing* [47] applied to 64 students studying Computer Science and Industrial Engineering, as part of an exploratory study. Both these areas of study include the Fundamentals of Computer Programming course in their curricula. These students used the CARAMBA tool for autonomous learning for a period of 3 months.

After using CARAMBA, we gave the students a questionnaire asking for their opinion on nine assertions. Table III shows these assertions according to three evaluation goals: *RS performance*, *user-centric effects*, and *learning effects* as suggested in [47]. Additional emphasis was put on evaluating both learning and user-centric effects (thus, more assertions are included for assessing these goals).

Two assertions related to the accuracy and response time of the system have been included (e.g., assertions A1 and A2). It is clear that both are less precise than those obtained from an off-line experiment, since they were measured based on the opinions of real users. However, we are aware that a greater number of technical tests will be needed and this will be the subject of future works.

The assertion A1 links the recommender system with the students programming skills refers to the fact that CARAMBA recommends exercises to the active user from the preferences of *other* users. However, this assertion was made under the assumption that the active user will be comfortable with CARAMBAs recommendations close to his/her current programming skills.

The nine exposed assertions were responded to using one of these five options: *strongly agree*, *agree*, *neither agree nor disagree*, *disagree*, and *strongly disagree*.

The results of the questionnaires (Figure 5) were organized into three groups: (a) Satisfaction of Computer Science students, (b) Satisfaction of Industrial Engineering students, and (c) Overall satisfaction. The last is the aggregation of the first two groups. A generally acceptable degree of satisfaction is appreciated. For instance, when more than 50% of the students have at least agreed with all the assertions.

However, differences exist between the first two groups. As expected, Computer Science students were more critical than the Industrial Engineering students (Figure 5-a and Figure 5-b). In both cases, a significant number of students did not agree with A1, indicating that more work had to be done regarding

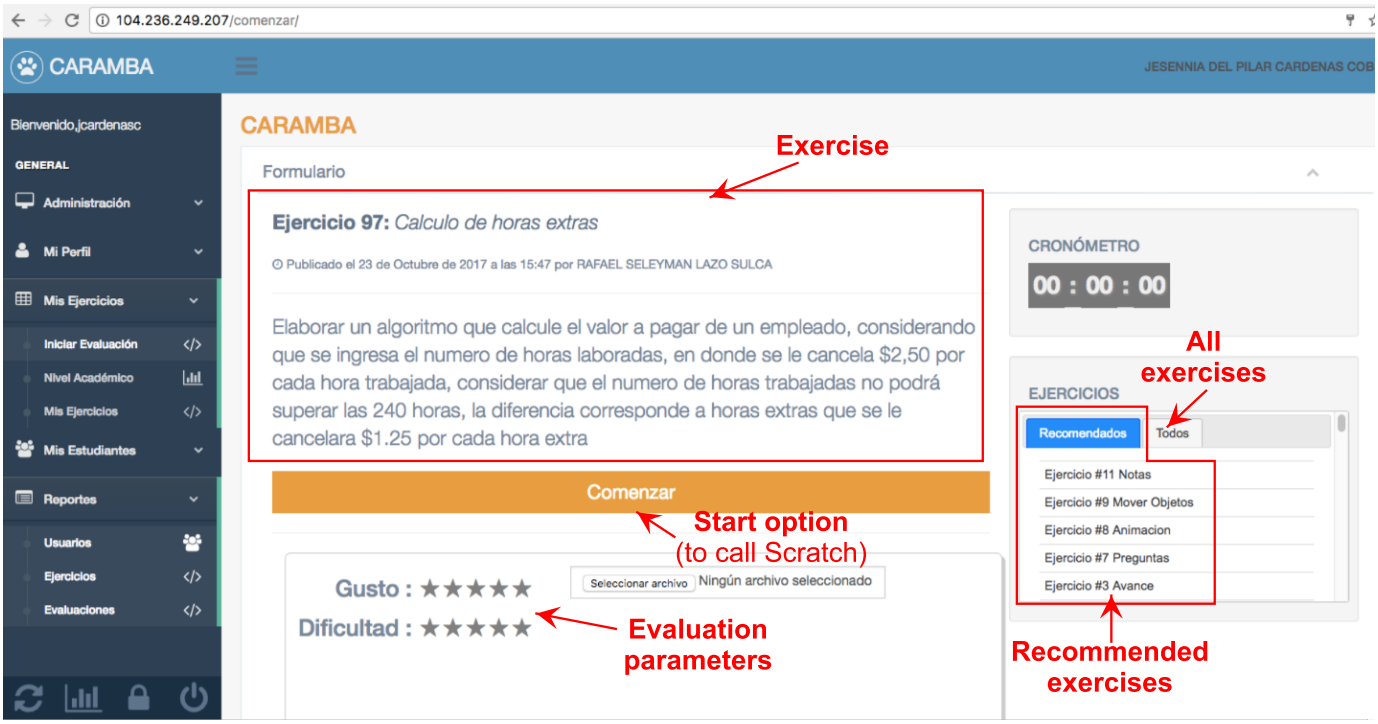

Fig. 3. CARAMBA showing the description of a selected exercise and its evaluation parameters.

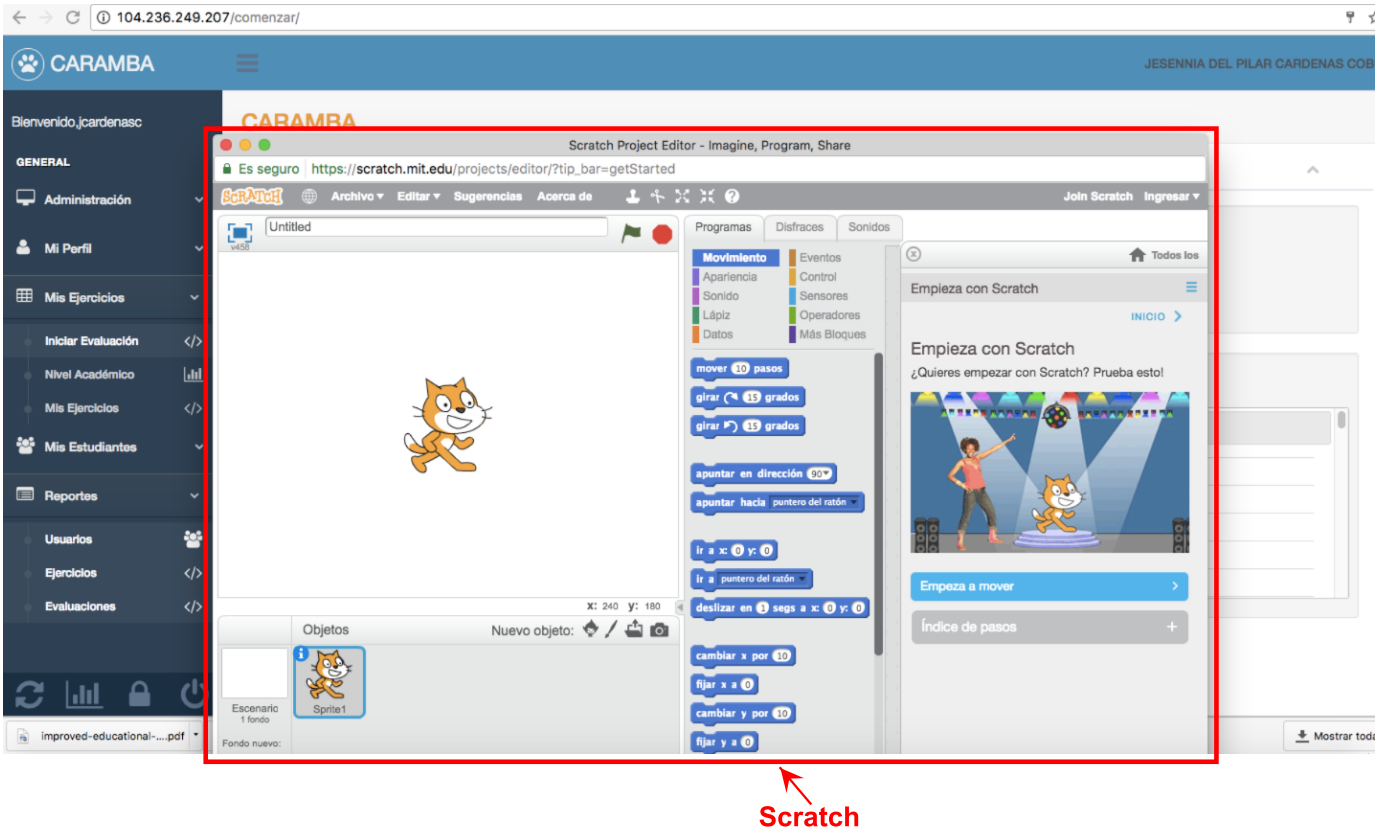

Fig. 4. Popup page of Scratch application inside CARAMBA.

TABLE II
MAIN STEPS OF THE RECOMMENDER SYSTEM INCLUDED IN CARAMBA.

Input: current user, u_k
Step 1. Build student profile u_k 1.1 Set of solved exercises(H_k) 1.2 Vector of assessments issued about H_k: Taste ($v_k^{(t)}$) and Complexity ($v_k^{(c)}$)
Step 2. For each student u_i registered in the system ($i \neq k$) ... 2.1. Build the student profile u_i - Select the solved exercises H_i by student u_i - Select the assessments issued about H_i: Taste ($v_i^{(t)}$) and Complexity ($v_i^{(c)}$) 2.2. Calculate similarity of assessments (only for exercises in common): $S^{(t)}(u_k,u_i) = \frac{v_i^{(t)} \cdot v_k^{(t)}}{\|v_k^{(t)}\| \cdot \|v_k^{(t)}\|}$ and $S^{(c)}(u_k,u_i) = \frac{v_i^{(c)} \cdot v_k^{(c)}}{\|v_k^{(c)}\| \cdot \|v_k^{(c)}\|}$ 2.3. Calculate the ratio of exercises in common: $S^{(p)}(u_k,u_i) = \frac{\lvert H_i \cap H_k \rvert}{\lvert H_k \rvert}$ 2.4. Calculate total similarity for u_i : $S(u_k,u_i) = S^{(t)} \cdot S^{(c)} \cdot S^{(p)}$
Step 3. Build a set Υ of students with $S > 0$.
Step 4. For each exercise e not evaluated by u_k. 4.1. Calculate the rated. $f_{u_k,e} = \bar{r}_{u_k} + \frac{\sum_{u \in \Upsilon}(r_{u_i,e} - \bar{r}_{u_i}) S(u_k,u_i)}{\sum_{u \in \Upsilon} S(u_k,u_i)}$
Step 5. Built a set E_k with n best rated exercises ($n <= 10$)
Step 6. If new exercises appear then add them to the E_k
Output: Recommended exercises, E_k

TABLE III
LIST OF ASSERTIONS EMPLOYED IN THE QUESTIONNAIRES FOR ASSESSING CARAMBA.

Evaluation Goal	Code	Assertion
RS performance	A1	CARAMBA recommends exercises to me according to my skills in computer programming.
	A2	The interaction with CARAMBA is fast enough.
User-centric effects	A3	I believe that *taste* and *complexity* are not only simple but also effective criteria for evaluating the exercises I faced.
	A4	CARAMBA presents a comfortable graphical user interface and navigation.
	A5	I consider CARAMBA to be a different but better system than Scratch without recommendations.
Effects of learning	A6	I have more chance of passing if I use CARAMBA.
	A7	I believe that CARAMBA will help me to improve my academic performance in the subject of Fundamentals of Computer Programming.
	A8	CARAMBA helps personalize my learning in computer programming.
	A9	I think that by using CARAMBA I have improved my autonomous learning.

system accuracy. A similar conclusion can be derived from the system time response (A2).

As for the user-centric assertions (A3, A4, and A5), more than 60% of the students from both courses at least agree and 40% strongly agree with assertion A5: that the proposed system is better.

Finally, regarding the assertions for evaluating the learning effects (A6, A7, A8, and A9), a clear difference exists between both areas of study. For instance, about 60% of Computer Science students agree with those indicators, while 75% of Industrial Engineering students agree.

In general, there was a suitable satisfaction level from the students of the proposed system (Figure 5-c).

B. Effect on learning

Motivated by the satisfactory conclusions obtained in the previous section, we decided to carry out a quantitative study to measure the effect of the CARAMBA tool on the concept of students. The experiment was carried out with 88 students enrolled in the CS1 course, taught in the first semester of the Computer Science Engineering program at Universidad Estatal de Milagro. Before attending CS1, the students received 64 hours of basic preparation in algorithms, corresponding to the pre-university stage[2]. The Structure of Programming Fundamentals is shown in Table IV.

The 88 students were randomly divided into three homogeneous groups according to the learning strategy, as shown in

[2]An intensive preparation course applied by the Ecuadorian Ministry of Education to compensate for the steep learning curve upon entering higher education.

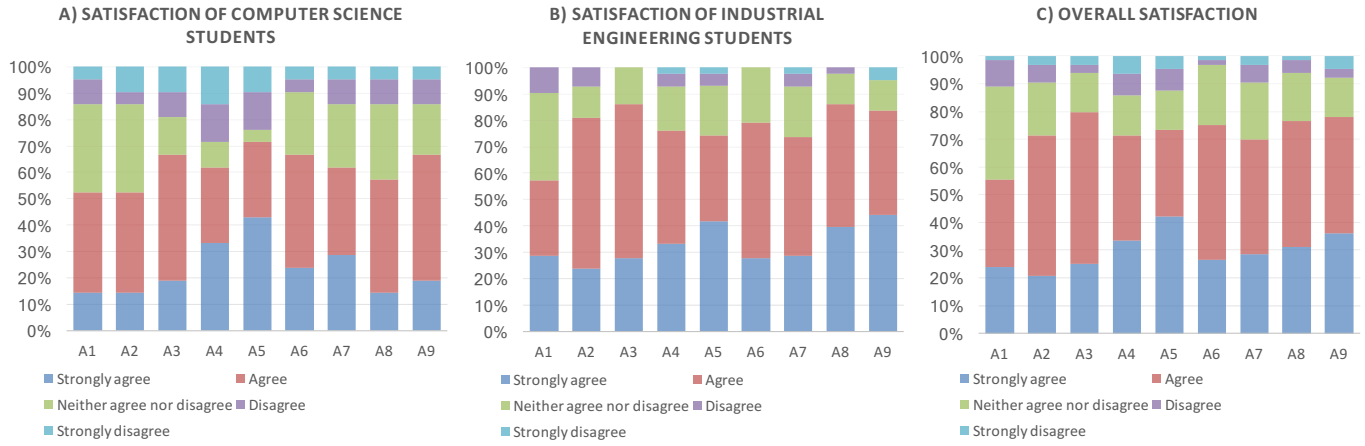

Fig. 5. Satisfaction of students with the system according to the questions.

TABLE IV
COURSE SYLLABUS BY WEEK.

Weeks	Programming Fundamentals
1-3	Pseudo-code, Flowchart, Variables, Arithmetic and Conditional Expressions
4-5	Cycles
6-7	Concurrency
8	Midterm exam
9-10	Divide and conquer
11	Lists
12	Sorting lists
13-16	Dynamic programming and complexity
17	Final Exam

TABLE V
STRUCTURE OF CONSIDERED GROUPS.

Group	Learning Strategy	Female	Male	Average Age
G1	Traditional	8	21	19.6
G2	Scratch-only	9	20	21.1
G3	CARAMBA	9	21	19.3

Table V. The same teachers were attending each group. To perform a fair comparison among the three groups, we employed the same type and quantity (109) of exercises in all three. Fig. 6 shows the distribution of the exercises according to the concept (Variable and Initialization, Cycles, Conditionals, and Concurrency). See that a similar distribution exists among the concepts, which provides enough diversity to the student. In order to avoid the Hawthorne effect [51], students and teachers were not informed about their participation in the study.

The study was organized to answer the following research questions:

- RQ1 (Concept Learning): To what extent will Scratch's learning strategy and an exercise recommendation system impact the students ability to learn programming basics?
- RQ2 (Final Performance): To what extent will Scratch's learning strategy and a exercise recommendation system impact the students' final performance in CS1?

For both questions (RQ1 and RQ2), we selected the learning strategy applied to each group as the *independent variable* (see Table V). The following is a brief explanation of what each

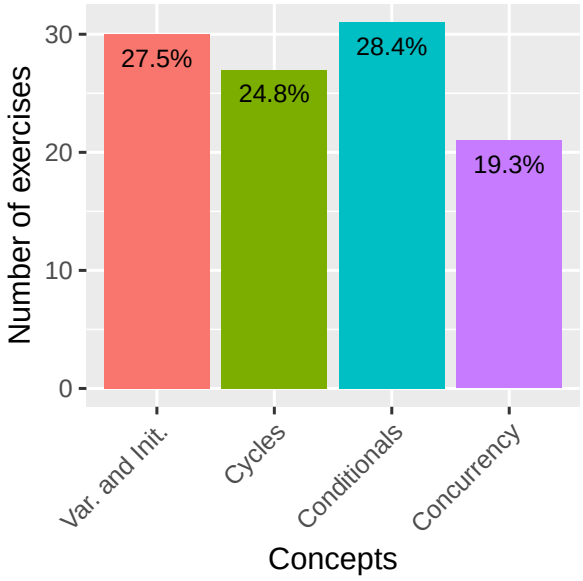

Fig. 6. Distribution of the number of exercises according to the concept.

consists of:

- Traditional: The teacher applies traditional teaching methods without using technology. The exercises used in the practices and independent study are proposed and controlled by the teacher.
- Scratch-only: The teacher uses the Scratch tool to intro-

TABLE VI
DESCRIPTIVE STATISTICS OF PRE-TEST RESULTS FOR EACH GROUP.

Groups	Mean	Variance	Standard Deviation
G1	3.24	2.86	1.69
G2	3.50	2.59	1.61
G3	3.46	3.01	1.73

duce and practice programming concepts. The exercises used in the independent study are proposed and controlled by the teacher.

- CARAMBA: As with the previous strategy, the teacher uses the Scratch tool to introduce and practice concepts in the classroom, but independent study is personalized by means of an exercise recommendation system.

Notice that our main goal is to find out whether the personalized way of facing exercises provided by the recommender system of CARAMBA has a significant impact on the learners. While traditional and Scratch-only approaches provided an organized path, where the exercises are revealed to students corresponding to the order of topics, with CARAMBA the students are free to build their own paths, by choosing to select the recommended exercises.

Below, we will discuss the results obtained for these research questions. In all cases, the assessment scale of the exams was 0-10, where 0 is the lowest grade and 10 is the highest.

1) RQ1 (Concept Learning): To provide answers to RQ1, we focused on the first eight weeks of the CS1 course. To measure the level of each group, an initial test (pre-test) was applied, taking into account four basic concepts: Variables, Cycles, Conditionals and Concurrency. At the end of week eight another test (post-test) was applied evaluating the same concepts.

The *dependent variables* for this RQ were the learning levels achieved by the students in each of the concepts. These levels were determined by ratings obtained in the post-test, applied after eight weeks of the experiment.

A similar knowledge level was assessed in the groups at the beginning of the study. Table VI shows the corresponding statistics for the groups in the pre-test. In these results, it can be seen that the mean values achieved by the three groups were low (on a scale of 0-10).

Table VII presents the descriptive statistics of the variables measuring the concept learning of the students (without distinction of group), both before and after applying the learning strategies. A previous analysis allowed us to determine:

- the strength of the applied learning strategies. This is because the highest mean values are reached after applying these strategies (in the post-test).
- that the greatest learning was obtained in the concept *Variable and initialization*, the difference between the pre-test and post-test was the largest.
- *Concurrency* is the most difficult concept to learn, because the values of the post-test were the lowest of all.

A preliminary reliability analysis across the Cronbach alpha revealed acceptable internal concordance within the allowed ranges (0.77). Alternately, the consistency study among evaluators from Kendall's W [52] also determined a positive result (w = 0.685) with a significance level of 6.33E-80.

Note that we applied both reliability and concordance tests to the overall dataset used for the subsequent statistical analysis, that is, involving all the evaluations performed during pretest and post-test stages and the three groups (traditional, Scratch-only, and Caramba). So, what we consider as scorers in our concordance test (Kendall w) were the students, but not the teachers.

a) Data analysis: In this study, we applied an experimental design of pre-test/post-test to the corresponding groups. Table VIII shows a descriptive analysis of the post-test variables depending on the learning strategy. It can be seen that the groups which used Scratch with the exercise recommendation system obtained the highest average values for each concept as well as the lowest values of standard deviation.

In order to determine if a learning strategy significantly influenced the students' cognitive development, we then applied an ANCOVA test by selecting results obtained in the pre-test as co-variables.

b) Results: For this study, four hypotheses were defined (one for each concept studied) as shown below:

- H1: Using the Scratch tool with an Integrated Exercise Recommendation System as a learning strategy has a significant impact on students identifying and initializing variables.
- H2: Using the Scratch tool with an Integrated Exercise Recommendation System as a learning strategy has a significant impact on students identifying and using different types of cycles.
- H3: Using the Scratch tool with an Integrated Exercise Recommendation System as a learning strategy has a significant impact on students identifying and using the different types of conditionals.
- H4: Using the Scratch tool with an Integrated Exercise Recommendation System as a learning strategy has a significant impact on students identifying and using Concurrency.

When applying the ANCOVA test (Table IX), we found that the results achieved by the learning strategies that were applied in each concept differed significantly $(Sig. < 0.05)$. Thus, we proceeded with post-hoc analysis to identify where these differences existed.

The Dunnett test represents a post-hoc alternative using a control treatment and comparing it with other treatments. In our case, we selected the results achieved by the CARAMBA learning strategy as base of comparison (control group in Dunnett's test) to compare it with the results of the other strategies.We selected this test because our hypotheses were enunciated to determine if the use of CARAMBA siginificantly improved the learning of programming languages.

Table X summarizes the values obtained by this test, which shows that:

- The CARAMBA strategy achieved significantly higher results than the traditional strategy $(Sig. < 0.05)$ in all concepts.

TABLE VII
DESCRIPTIVE STATISTICS

Dependent variable		Average	Standard Deviation	N	Min	Max
Variable and Initialization	Pre-test	3.89	1.35	88	1	7
	Post-test	7.03	1.70	88	3	10
Cycles	Pre-test	3.42	1.47	88	1	7
	Post-test	6.56	2.15	88	1	10
Conditionals	Pre-test	3.77	1.88	88	1	8
	Post-test	6.71	1.90	88	1	10
Concurrency	Pre-test	2.35	1.12	88	1	5
	Post-test	5.45	2.13	88	1	8

TABLE VIII
DESCRIPTIVE STATISTICS FOR POST-TEST VARIABLES

Variable	Teaching method	Average	Standard Deviation	N
Post-test Variable and initialization	Traditional method	5.86	1.60	29
	Scratch	7.21	1.40	29
	CARAMBA	8.00	1.31	30
	Total	7.03	1.71	88
Post-test Cycles	Traditional method	5.17	2.51	29
	Scratch	6.66	1.50	29
	CARAMBA	7.80	1.45	30
	Total	6.56	2.15	88
Post-test Conditionals	Traditional method	5.45	2.11	29
	Scratch	7.03	1.64	29
	CARAMBA	7.80	1.03	30
	Total	6.77	1.90	88
Post-test Concurrency	Traditional method	4.52	2.03	29
	Scratch	5.38	2.08	29
	CARAMBA	6.43	1.89	30
	Total	5.45	2.13	88

TABLE IX
RESULTS OF THE ANCOVA TEST

Independent variable	Dependent variable	Quadric	F	Sig.
Learning	Variable and Initializations	34.35	15.85	5.12E-5
	Cycles	51.11	14.51	1.15E-4
	Conditionals	42.26	15.55	1.02E-4
	Concurrency	27.19	6.81	3.21E-3

- The CARAMBA strategy also achieved significantly higher results than the Scratch strategy but only in concepts of variables and initialization and that of Cycles and Concurrency. In cases of conditional concepts, the differences found were not significant ($Sig. > 0.05$).

2) RQ2 (Final Performance): In this RQ2, we focused on the students' final grades, which were calculated by adding together the three evaluations of the course: cumulative grade up to week eight, cumulative grade up to week 16, and final exam grade.

Thus, the level of knowledge acquired in the subject by the student was considered as the *dependent variable*. The values of this variable, as mentioned above, represented the final grade of each student.

a) Data analysis: In this case, a comparative analysis was applied between groups to identify whether the learning strategy had a significant impact on students advancement.

Analyzing the descriptive statistics presented in Table XI, it can be seen that the group who used the CARAMBA tool was the one with the highest average advancement value (7.5083) and the lowest standard deviation (0.92). This result shows that this was the only group whose general average exceeded the threshold allowed to pass a subject (greater than or equal to 7).

We will then apply a one-factor ANOVA analysis to determine if the differences between the means are significant.

b) Results: In this study, the following research hypotheses were defined:

- H5: Using the Scratch tool with an Integrated Exercise Recommendation System as a learning strategy has a significant impact on the final class results.

TABLE X
RESULTS OF THE DUNNETT TEST FOR EACH CONCEPT.

Control(j)	Treatments(i)	Concepts	Diff. Means (i-j)	Sig.
CARAMBA	Traditional methods	Variables and initialization	-2.14	2.76E-7
		Cycles	-2.63	6.54E-7
		Conditionals	-2.35	4.22E-7
		Concurrency	-1.92	3.98E-4
	Scratch	Variables and initialization	-0.79	0.04
		Cycles	-1.14	0.02
		Conditionals	-0.77	0.07
		Concurrency	-1.05	0.04

TABLE XI
DESCRIPTIVE STATISTICS BY LEARNING STRATEGY

Teaching method	Average	Standard Deviation	N	Min	Max
Traditional method	5.25	1.31	29	3.00	7.75
Scratch	6.57	1.29	29	3.50	9.00
CARAMBA	7.51	0.92	30	3.00	9.50
Total	6.45	1.50	88	3.00	9.50

TABLE XII
RESULTS OF THE DUNNETT TEST FOR ACADEMIC PERFORMANCE

Control (j)	Treatment(i)	Diff. Means (i-j)	Sig.
CARAMBA	Traditional method	-2-26	3.46E-7
	Scratch	-0.94	0.003

The ANOVA test of an executed factor determined the existence of significant differences between the means of the 3 groups (p value = 8.838E-10). Furthermore, the Dunnett multiple comparison test (Table XII) showed that the control group achieved significantly better results than other treatments in the study $(Sig. < 0.05)$.

In another analysis, we calculated the pass rate, or percentage of students who passed in the group that used the CARAMBA tool, and compared it with the historic results of the Fundamentals of Programming course. In Figure 7, it can be observed that: it was the first time that the pass rate was over 52 %, the value reached was 8% higher than that of the 2014-2015 (2nd) period pass rate (which was the highest value prior to this study) and 21% higher than the program's historical average (30.38 %).

C. Analysis of CARAMBA utilization

In this Section we present evidences supporting how assisted learning using CARAMBA helps students to significantly improve their results. Concretely, we focus on utilization, which is the level of practice of the students with the exercises.

Concretely, we will try to answer the following questions:

- Q1: Did students actually use CARAMBA?
- Q2: Did students practice by using CARAMBA recommended exercises?
- Q3: How does doing sets of concept learning problems organized in levels help students to pass the course of CS1?
- Q4: How did CARAMBA help students to pass the course of CS1?

To answer Q1, we analyzed the number of exercises done by students. Figure 8 shows the average number of exercises done by the students in the 16-week course. We see that the exercise rate per student-week remained above 4 throughout the course. Note that the tendency (continuous line) of this rate fluctuated across the course and had an error of approximately one exercise.

The optimal rate is 6.81 which is calculated from the total number of exercises available (109) per student in the 16 weeks, assuming that each student would ideally complete the same number of exercises per week. We observe that the values obtained by the students represented more than the 58%. This is not only an affirmative answer to question 1, but also indicates that the level of practice with CARAMBA was important.

To answer the second question we evaluate the students' interactions with the exercices. Figure. 9 shows the distribution of the students' interactions with the exercises included in CARAMBA, taking into account the origin of the exercise (Pool of exercises and Recommender system). Please note here that an exercise can be selected by the student in two ways using CARAMBA: from the set of all available exercises (i.e. without recommendation) or from the set created by the recommendation system. Firstly, Fig. 9, shows that most of the interactions (about 80%) of the students with the system were due to the interest in the exercises recommended by CARAMBA. Secondly, only 20% are interactions that came out of non-recommended exercises. This means that the students practiced with CARAMBA mostly through the recommendation system.

Questions Q3 and Q4 are closely related. Both questions depend on the definition of the system's level of use system. In this case, we have considered that the use of CARAMBA by a student can be defined as the percentage of exercises performed from those totally available (regardless of origin: recommended or not). Specifically, for question Q3 this percentage was calculated on the basis of the number of exercises

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TLT.2019.2901457, IEEE Transactions on Learning Technologies

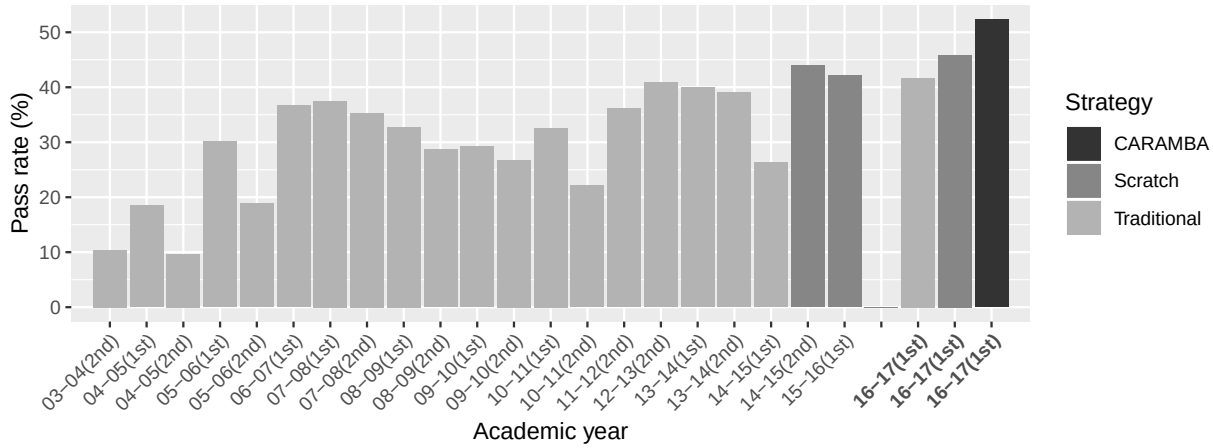

Fig. 7. Historical pass rate of UNEMI. The semester where the present study took place is denoted as 16–17(1st) in boldface.

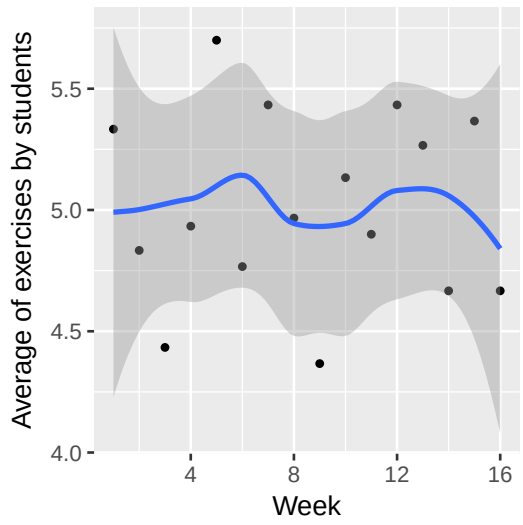

Fig. 8. Rate of exercises evaluated per student/week (points). The continuous line correspond to the conditional mean using a Local Polynomial Regression Fitting. The shaded area around the line correspond to the confidence interval (level = 0.95).

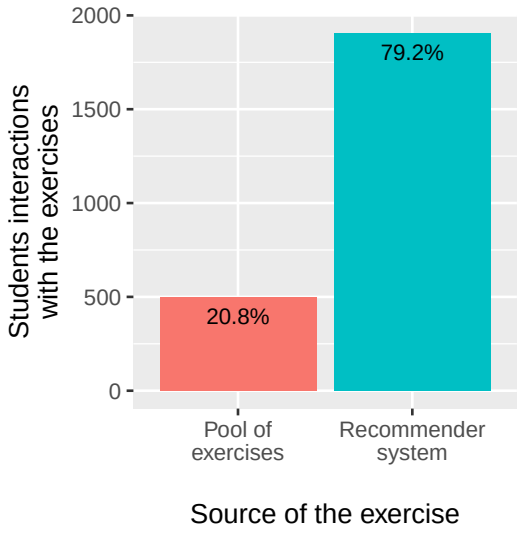

Fig. 9. Distribution of the type of students' interactions with CARAMBA according to source: *Pool of exercises* (without recommendation) and *Recommender system*

available for each concept. Thus, each student would have five levels of use associated for each concept (Variable and Initialization, Cycles, Conditionals, and Concurrency) and one overall (taking into account all available exercises). A higher value for the level of use indicated that the students interacted with a lot of exercises, while a low value meant the opposite.

In Fig. 10, the level of use of the system (x-axis) and the final grade of the students (y-axis) were summarized for each concept. The grade is in the range of 0 to 10, where 0 is the worst possible grade and 10 the best. To facilitate the visual comparison in this graph, we have identified each student with a category according to their grade: failed students (FAILED with a grade below 7), average students (Average with a grade between 7 and 8.5), and finally, outstanding students (Excellent). Figure. 10 allows us to draw some important conclusions. Initially, it can be seen that the students had usage levels above 50% in all concepts. It is interesting to see that there was one student who used all the exercises of the concept in the Concurrency level case. However, the most important conclusion here is that for at least three of the concepts (Cycles, Conditionals, and Concurrency) there is a correlation between the level of use of the system and the student's success/failure. For instance, one can see that excellent and average students are associated with high usage levels. In the case of the failed students, the great dispersion between the levels of student utilization does not allow us to identify this relationship.

We have proceeded in a similar manner for Q4. Figure 11 shows the same analysis but considers the overall utilization level of each student. In this plot, the pattern of correlation (positive) is much better appreciated.

To support our perception that the level of use of CARAMBA has a positive/negative influence on the student's success/failure in the subject, we have proceeded with a correlation test. Here, we have used the Pearson's correlation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TLT.2019.2901457, IEEE Transactions on Learning Technologies

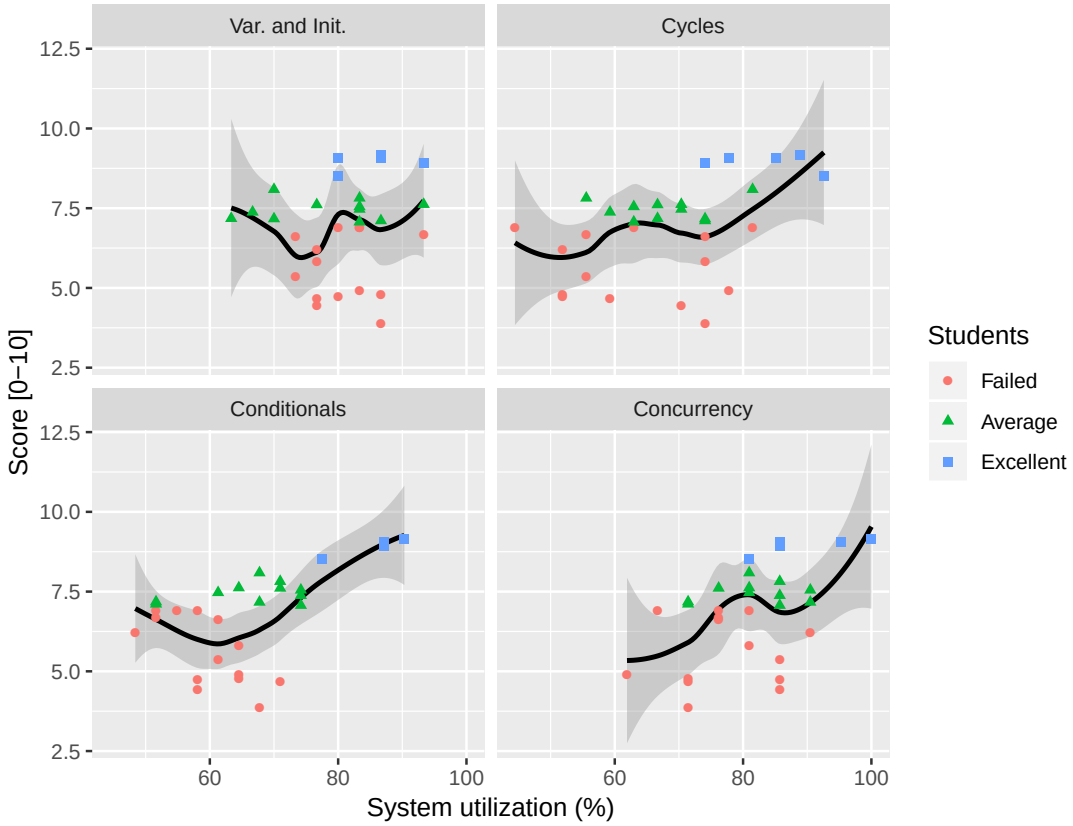

Fig. 10. System utilization vs. Score according to the Concept (Var. and Init., Cycles, Conditionals, and Concurrency). Each point represents an student of a certain group (see legend). The continuous line corresponds to the conditional mean using a Local Polynomial Regression Fitting, where the shaded area around the line corresponds to the confidence interval (level = 0.95).

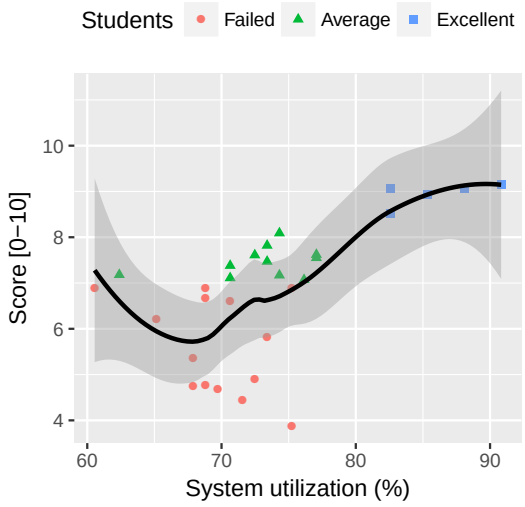

Fig. 11. Overall system utilization vs. Score. Each point represents a student of a certain group (see legend). The continuous line corresponds to the conditional mean using a Local Polynomial Regression Fitting, where the shaded area around the line corresponds to the confidence interval (level = 0.95).

test, which was applied by using the level of use of the system by each student and the corresponding grade as variables. The results shown in the Table XIII consider each concept and all exercises (overall). Please note here that the p-value of the test, the correlation coefficient, and the minimum and maximum values of the confidence interval (for 95%) have been included.

From Table XIII it can be concluded that there is a positive correlation between the level of use and the student's grade. However, in the case of variable and initialization, this correlation is not significant ($p - value > 0.05$). Questions Q3 and Q4 can therefore be answered in an affirmative way.

TABLE XIII
PEARSON'S PRODUCT-MOMENT CORRELATION TEST FOR SYSTEM UTILIZATION VS. SCORES.

Concept	p-value	Correlation coefficient	Min. C.I*.	Max. C.I
Var. and Init.	5.35E-01	1.18E-01	-2.53E-01	4.59E-01
Cycles	**1.35E-02**	4.46E-01	1.02E-01	6.95E-01
Conditionals	**1.68E-03**	5.49E-01	2.35E-01	7.59E-01
Concurrency	**6.94E-03**	4.82E-01	1.48E-01	7.18E-01
Overall	**1.27E-04**	3.43E-01	1.74E-01	4.92E-01

*95% confidence interval

We can thus conclude that the primary cause of CARAMBA's success is the high level of practice achieved by

the students with the tool. However, an unanswered question is what causes students to exhibit such a high level of practice with CARAMBA. In our opinion, this is mainly caused since students are very motivated with CARAMBA. We have observed that today's students, like most people in modern society, are linked to information and communication technologies. Thus, we assume that by using a tool like CARAMBA, which allows students to learn autonomously, students would feel more motivated. Consequently, this motivation would lead to a higher level of practice, which would lead to a higher probability of success in the subject. We recognize that such an assumption implies an in-depth formal study that explains these and other issues.

V. Future Work

In this Section we present the future work related to this research.

- Recommendation of controlled exercises: We will study whether building recommendations take into account the order in which the topics are taught to improve students' learning. To this end, we will study content-based recommendation systems. These systems recommend items with similar characteristics to those already evaluated by a user.
- Scalability of the system: This tool is being adapted to other study scenarios with more users and exercises. In this sense, we are studying the incorporation of a collaborative filtering system based on the [53] model. This type of system reduces the size of the valuation matrix by transforming it into characteristics that represent common factors present in the original matrix and allow the system to recognize patterns, which may be hidden in the data set.
- CARAMBA Functionality: Incorporating other statistical details and improving the user interface.

VI. Conclusion

In this study we present CARAMBA, an easy-to-use Web Application involving Scratch alongside a recommender system for exercises. Our goal has been to enhance the learning of computer programming at the college level. The experimental study was developed in two different stages and periods:

- In the first stage, a group of computer science and engineering students used the application for independent study. These students were asked to assess nine indicators regarding three goal areas: (1) recommender system performance; (2) user-centric effects; and (3) learning effects. In general, a significant level of satisfaction among the students was observed.
- In the second stage, we conducted a quantitative study regarding concept learning and academic performance in CS1 students. For this, 3 learning methodologies were applied: the traditional method, Scratch only, and the CARAMBA application. The results are as follows:
  - Concept learning: The study confirmed that Scratch with an exercise recommendation system significantly enhances the learning of basic programming concepts. This tool provided a customized environment for the study, which helped to develop autonomous learning amongst students based on their cognitive preferences. Statistical analysis allowed us to validate the formulated hypotheses (H1, H2, H3 and H4), showing that the effect of the proposal was significantly superior than other applied strategies.
  - Final performance: The performance study statistically corroborated with the H5 research hypothesis, showing that the learning strategy significantly influences rates of student advancement. The results showed that the group which used the Scratch tool with the exercise recommendation system obtained the highest scores among all experimental groups.
  - CARAMBA utilization: results show that there is a significant positive correlation between the high level of practice with CARAMBA and the possibility of success in the subject. This is consistent with the study on cognitive learning and performance, which points out that students feels motivated when using CARAMBA.

Additionally, we observed that the pass rate of students using CARAMBA surpassed the historic pass rate for the subject of Fundamentals of Programming. Concretely, the pass rate achieved by our proposal was over 52%, which is 8% higher than the rate achieved during a previous experience using only Scratch (without recommendation) and 21% higher than the historical results of tradition teaching (without Scratch).

Lastly, we expected that our findings would increase researchers interest in this topic. Extending Scratch with suitable ICT progresses to enhance programming learning in college students seems to be a promising alternative to traditional approaches of teaching. We believe that todays students need technology-based learning strategies. Our future work will be oriented for developing and assessing these technological solutions.

Acknowledgments

This research has been conducted during the development of the project FOCICYT: Soft Computing Applications in Higher Education Environments, which is supported by the Universidad Técnica Estatal de Quevedo (2017-2018).

References

[1] I. Huet, O. Pacheco, J. Tavares, and G. Weir, "New challenges in teaching introductory programming courses: a case study," in *34th Annual Frontiers in Education, 2004. FIE 2004.* IEEE, 2004, pp. 286–290.

[2] H. C. Jiau, J. C. Chen, and K.-F. Ssu, "Enhancing Self-Motivation in Learning Programming Using Game-Based Simulation and Metrics," *IEEE Transactions on Education*, vol. 52, no. 4, pp. 555–562, nov 2009. [Online]. Available: http://ieeexplore.ieee.org/document/5164890/

[3] T. M. Connolly, M. Stansfield, and T. Hainey, "An application of games-based learning within software engineering," *British Journal of Educational Technology*, vol. 38, no. 3, pp. 416–428, 2007.

[4] D. Tsompanoudi, M. Satratzemi, and S. Xinogalos, "Evaluating the Effects of Scripted Distributed Pair Programming on Student Performance and Participation," *IEEE Transactions on Education*, vol. 59, no. 1, pp. 24–31, feb 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7089313/

[5] S. K. Andrianoff, D. B. Levine, S. K. Andrianoff, and D. B. Levine, "Role playing in an object-oriented world," *ACM SIGCSE Bulletin*, vol. 34, no. 1, p. 121, mar 2002. [Online]. Available: http://portal.acm.org/citation.cfm?doid=563517.563386

[6] J.-M. Sáez-López, M. Román-González, and E. Vázquez-Cano, "Visual programming languages integrated across the curriculum in elementary school: A two year case study using scratch in five schools," *Computers and Education*, vol. 97, pp. 129 – 141, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131516300549

[7] S. Cooper, W. Dann, R. Pausch, S. Cooper, W. Dann, and R. Pausch, "Teaching objects-first in introductory computer science," in *Proceedings of the 34th SIGCSE technical symposium on Computer science education - SIGCSE '03*, vol. 35, no. 1. New York, New York, USA: ACM Press, 2003, p. 191. [Online]. Available: http://portal.acm.org/citation.cfm?doid=611892.611966

[8] M. C. Carlisle, T. A. Wilson, J. W. Humphries, and S. M. Hadfield, "RAPTOR: Introducing Programming to Non-majors with Flowcharts," *J. Comput. Sci. Coll.*, vol. 19, no. 4, pp. 52–60, 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1050231.1050238

[9] C. J. Bouras, V. Poulopoulos, and V. Tsogkas, "Squeak Etoys: Interactive and Collaborative Learning Environments," in *Handbook of Research on Social Interaction Technologies and Collaboration Software: Concepts and Trends*, T. Dumova and R. Fiordo, Eds. Hershey, PA, USA: IGI Global, 2010, pp. 417–427. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60566-368-5.ch037

[10] D. J. Malan, H. H. Leitner, D. J. Malan, and H. H. Leitner, "Scratch for budding computer scientists," in *Proceedinds of the 38th SIGCSE technical symposium on Computer science education - SIGCSE '07*, vol. 39, no. 1. New York, New York, USA: ACM Press, 2007, p. 223.

[11] J. Maloney, M. Resnick, and N. Rusk, "The Scratch programming language and environment," *ACM Transactions on Computing Education*, vol. 10, no. 4, pp. 1–15, 2010.

[12] I. F. de Kereki, "Scratch: Applications in Computer Science 1," in *2008 38th Annual Frontiers in Education Conference*. IEEE, oct 2008, pp. T3B–7–T3B–11. [Online]. Available: http://ieeexplore.ieee.org/document/4720267/

[13] S. Mishra, S. Balan, S. Iyer, and S. Murthy, "Effect of a 2-week Scratch Intervention in CS1 on Learners with Varying Prior Knowledge," in *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ser. ITiCSE '14. New York, NY, USA: ACM, 2014, pp. 45–50. [Online]. Available: http://doi.acm.org/10.1145/2591708.2591733

[14] U. Wolz, H. H. Leitner, D. J. Malan, J. Maloney, U. Wolz, H. H. Leitner, D. J. Malan, and J. Maloney, "Starting with scratch in CS 1," in *Proceedings of the 40th ACM technical symposium on Computer science education - SIGCSE '09*, vol. 41, no. 1. New York, New York, USA: ACM Press, 2009, p. 2. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1508865.1508869

[15] E. Tanrikulu and B. C. Schaefer, "The users who touched the ceiling of scratch," in *World Conference on Educational Technology Researches - 2011*, H. . Yalin, F. Adiloglu, H. Boz, S. Karata, and F. Ozdaml, Eds., vol. 28, 2011, pp. 764–769.

[16] J. Cárdenas-Cobo, P. Novoa-Hernández, A. Puris, and D. Benavides, *Recommending Exercises in Scratch: An Integrated Approach for Enhancing the Learning of Computer Programming*. Cham: Springer International Publishing, 2018, pp. 255–271.

[17] G. Fesakis and K. Serafeim, "Influence of the familiarization with "scratch" on future teachers' opinions and attitudes about programming and ICT in education," *ACM SIGCSE Bulletin*, vol. 41, no. 3, p. 258, 2009.

[18] S. Garner, "Learning to program from scratch," in *Proceedings - 2009 9th IEEE International Conference on Advanced Learning Technologies, ICALT 2009*, 2009, pp. 451–452.

[19] L. A. Vaca-Cárdenas, F. Bertacchini, A. Tavernise, L. Gabriele, A. Valenti, D. E. Olmedo, P. Pantano, and E. Bilotta, "Coding with scratch: The design of an educational setting for elementary pre-service teachers," in *2015 International Conference on Interactive Collaborative Learning (ICL)*, Sept 2015, pp. 1171–1177.

[20] ——, "Coding with scratch: The design of an educational setting for elementary pre-service teachers," in *2015 International Conference on Interactive Collaborative Learning (ICL)*. IEEE, sep 2015.

[21] L. A. Vaca-Cárdenas, A. Tavernise, F. Bertacchini, L. Gabriele, A. Valenti, P. Pantano, and E. Bilotta, "An educational coding laboratory for elementary pre-service teachers: A qualitative approach," *International Journal of Engineering Pedagogy (iJEP)*, vol. 6, no. 1, p. 11, feb 2016.

[22] J. A. Martínéz-Valdés, J. Angél Vélazquéz-Iturbidé, and R. Hijon-Néira, "A (relatively) unsatisfactory experience of use of Scratch in CS1," in *ACM International Conference Proceeding Series*, vol. Part F132203, 2017.

[23] J. Moreno-León and G. Robles, "Dr. scratch: A web tool to automatically evaluate scratch projects," in *Proceedings of the Workshop in Primary and Secondary Computing Education*, ser. WiPSCE '15. New York, NY, USA: ACM, 2015, pp. 132–133.

[24] J. Moreno-León, G. Robles, and M. Román-González, "Towards data-driven learning paths to develop computational thinking with scratch," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[25] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*. Boston, MA: Springer US, 2011, ch. Introduction to Recommender Systems Handbook, pp. 1–35.

[26] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutierrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[27] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: A survey," *Decision Support Systems*, vol. 74, pp. 12 – 32, 2015.

[28] C. Desrosiers and G. Karypis, *A Comprehensive Survey of Neighborhood-based Recommendation Methods*. Springer US, 2011, ch. Recommender Systems Handbook, pp. 107–144.

[29] M. Elahi, F. Ricci, and N. Rubens, "A survey of active learning in collaborative filtering recommender systems," *Computer Science Review*, vol. 20, pp. 29–50, 2016.

[30] N. Manouselis, H. Drachsler, R. Vuorikari, H. Hummel, and R. Koper, *Recommender Systems in Technology Enhanced Learning*. Boston, MA: Springer US, 2011, pp. 387–415.

[31] N. Manouselis, H. Drachsler, K. Verbert, and E. Duval, *Survey and Analysis of TEL Recommender Systems*. New York, NY: Springer New York, 2013, pp. 37–61.

[32] A. Klašnja-Milićević, M. Ivanović, and A. Nanopoulos, "Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions," *Artificial Intelligence Review*, vol. 44, no. 4, pp. 571–604, 2015.

[33] R. Farzan and P. Brusilovsky, *Social Navigation Support in a Course Recommendation System*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 91–100.

[34] M. Khribi, M. Jemni, and O. Nasraoui, "Automatic recommendations for e-learning personalization based on web usage mining techniques and information retrieval," *Educational Technology & Society*, vol. 12, no. 4, pp. 30–42, 2009.

[35] X. Wan and T. Okamoto, "Utilizing learning process to improve recommender system for group learning support," *Neural Computing and Applications*, vol. 20, no. 5, pp. 611–621, 2011.

[36] X. Wan, Q. Jamaliding, and T. Okamoto, "Analyzing learners' relationship to improve the quality of recommender system for group learning support," *Journal of Computers*, vol. 6, no. 2, pp. 254–262, 2011.

[37] O. C. Santos and J. G. Boticario, "Requirements for semantic educational recommender systems in formal e-learning scenarios," *Algorithms*, vol. 4, no. 2, pp. 131–154, 2011.

[38] K. Ghauth and N. Abdullah, "The effect of incorporating good learners' ratings in e-learning contentbased recommender system," *Educational Technology and Society*, vol. 14, no. 2, pp. 248–257, 2011.

[39] G. Lee, M. Salehi, and I. N. Kmalabadi, "International conference on future computer supported education, august 22- 23, 2012,fraser place central - seoul a hybrid attributte based recommender system for e-learning material recommendation," *IERI Procedia*, vol. 2, pp. 565–570, 2012.

[40] P. Dwivedi and K. K. Bharadwaj, "Effective trust-aware E-learning recommender system based on learning styles and knowledge levels," *Educational Technology and Society*, vol. 16, no. 4, pp. 201–216, 2013.

[41] A. S. Tewari, A. Saroj, and A. G. Barman, "E-learning recommender system for teachers using opinion mining," *Lecture Notes in Electrical Engineering*, vol. 339, pp. 1021–1029, 2015.

[42] A. Ruiz-Iniesta, G. Jiménez-Díaz, and M. Gómez-Albarrán, "A framework for the rapid prototyping of knowledge-based recommender systems in the learning domain," *Journal of Research and Practice in Information Technology*, vol. 44, no. 2, pp. 167–181, 2012.

[43] J. Buder and C. Schwind, "Learning with personalized recommender systems: A psychological view," *Computers in Human Behavior*, vol. 28, no. 1, pp. 207–216, 2012.

[44] K. Peiris and R. c. Gallupe, "A Conceptual Framework for Evolving, Recommender Online Learning Systems," *Decision Sciences Journal of Innovative Education*, vol. 10, no. 3, pp. 389–412, 2012.

[45] A. R. Anaya, M. Luque, and T. García-Saiz, "Recommender system in collaborative learning environment using an influence diagram," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7193–7202, 2013.
[46] D. Leony, H. Parada Gélvez, P. Mnoz-Merino, A. Pardo, and C. Kloos, "A generic architecture for emotion-based recommender systems in cloud learning environments," *Journal of Universal Computer Science*, vol. 19, no. 14, pp. 2075–2092, 2013.
[47] M. Erdt, A. Fernández, and C. Rensing, "Evaluating Recommender Systems for Technology Enhanced Learning: A Quantitative Survey," *IEEE Transactions on Learning Technologies*, vol. 8, no. 4, pp. 326–344, 2015.
[48] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, Jan. 2004.
[49] M. Bramer, *Introduction to Data Mining*. London: Springer London, 2013, pp. 1–8.
[50] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. in Artif. Intell.*, vol. 2009, pp. 4:2–4:2, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/421425
[51] R. Spano, "Observer behavior as a potential source of reactivity: Describing and quantifying observer effects in a large-scale observational study of police," *Sociological Methods & Research*, vol. 34, no. 4, pp. 521–553, 2006.
[52] J. Kendall, "A new measure of rank correlation." *Biometrika*, vol. 30, pp. 81 – 93, 1938.
[53] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

Jesennia Cárdenas obtained the B.S. degree in Information Systems from the Escuela Politcnica del Litoral (ESPOL, Ecuador). She also has a Diploma in Higher Education by Competences from the Technical University of Ambato (Ecuador) and MSc. in Business Administration from the Business Technological University of Guayaquil (Ecuador). She is currently a Ph.D. candidate in Software Engineering from the University of Seville (Spain). She is a full-time professor and Dean of the Faculty of Engineering Sciences of the State University of Milagro (UNEMI, today). She has more than 16 years of professional experience in higher education. Her main research interests include software products and artificial intelligence applied to engineering education.

Amilkar Puris is Associate Professor and Researcher of the Universidad Estatal de Milagro (Ecuador,2016-2018) and the Universidad Técnica Estatal de Quevedo (Ecuador, 2013). He obtained a B.S. degree in computer science from the Universidad Central de Las Villas (Cuba,2004). He also received M.Sc. and Ph.D. degrees from the same institution in 2006 and 2010, respectively. His research interest involve metaheuristics, data science, evolutionary computation, and decision making in complex scenarios. He has had considerable experience in teaching computer programming and operation research at the college level.

Pavel Novoa-Hernández received a B.S. degree in Computer Science Engineering from the University of Holguin (Cuba) in 2007, and the Ph.D. degree in Information and Communication Technologies from the University of Granada (Spain) in 2013. Currently, he is with the Universidad Técnica Estatal de Quevedo (Ecuador), where he is an Associate Professor and researcher. In this institution, he is the lecturer of Discrete Mathematics and Computer Programming. He also collaborates with the Universidad Estatal de Milagro as a guest lecturer. His research interests involve soft computing, metaheuristics, dynamic optimization, and problem-solving in higher education environments.

Jos A. Galindo He has developed his professional activity in the United States, France, and Spain. His research areas are product lines software and the configuration of such products. He obtained a Ph.D. from the University of Seville and the University of Rennes 1 in March 2015 with honors and receiving the award for the best national thesis by SISTEDES. He has developed his post-doctoral research activity at INRIA, France. Currently working as a Juan de la Cierva researcher at the University of Seville, he continues his line of research on configuration, testing, and the evolution of highly configurable systems.

David Benavides is the Associate Professor at the University of Seville from 2010. He obtained a B.S. degree in Information Systems in 2000 from the Institute Superieur d'Electronique de Paris, France. He is an M.Sc. in Computer Engineering from the University of Seville, Spain (2001) and a Ph.D. in Software Engineering, University of Seville, Spain (2007). His main research interests include software product line and artificial intelligence applied to engineering education.