Recommending Exercises in Scratch: An Integrated Approach for Enhancing the Learning of Computer Programming

Chapter · January 2018

DOI: 10.1007/978-3-319-60937-9_20

4 authors:

Jesennia Cardenas
State University of Milagro
9 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Pavel Novoa-Hernández
Universidad Técnica Estatal de Quevedo
40 PUBLICATIONS **109** CITATIONS
SEE PROFILE

Amilkar Puris
Universidad Técnica Estatal de Quevedo
37 PUBLICATIONS **206** CITATIONS
SEE PROFILE

David Benavides
Universidad de Sevilla
100 PUBLICATIONS **3,288** CITATIONS
SEE PROFILE

Recommending exercises in Scratch: an integrated approach for enhancing the learning of computer programming

Jesennia Cárdenas-Cobo, Pavel Novoa-Hernández, Amikar Puris, and David Benavides

Abstract In this chapter we focused on how to improve the learning of computer programming in college students. From the reported success, we relied on Scratch, a visual programming language for enhancing the informal learning. Despite the progress achieved in the past, we have observe some issues regarding the use of Scratch by college students. First, there is a gap between the employed learning approaches since professors are constrained to classroom activities. Second, certain students feel unmotivated because they are confronted with programming exercises that do not fulfill their individual expectations. So, in order to solve this issue we propose an integrated approach consisting of a simple Web Application that includes Scratch as a project editor along with an Recommender System for exercises. The preliminary results demonstrate the positive impact of our proposal.

1 Introduction

In traditional education approaches (e.g., schools), professors play an important role because they guide the learning process according to a considered structure of the

J. Cárdenas-Cobo
Universidad Estatal de Milagro, Guayas, Ecuador.
Ciudadela Universitaria Km. 1.5 vía Km. 26
Tel.: +593 4 2715187
e-mail: jcardenasc@unemi.edu.ec

P. Novoa-Herández, A. Puris
Universidad Técnica Estatal de Quevedo, Los Ríos, Ecuador.
Guest lecturers at Universidad Estatal de Milagro, Ecuador.
e-mail: pnovoa@uteq.edu.ec, apuris@uteq.edu.ec

D. Benavides
Department of Computer Languages and Systems
Universidad de Sevilla, E.T.S.I. Informática, Av. Reina Mercedes s/n, 41012, Seville, Spain
e-mail: benavides@us.es

subject. In other words, the knowledge is revealed to the student in an organized way. This is frequently called formal learning (European Centre for the Development of Vocational Training, 2014). Regardless of the benefits of such an educational model, over the last 20 years the so- called Technology Enhanced Learning (TEL) has raised alternative models. TEL has been defined as an application domain that covers technologies for supporting all forms of teaching and learning activities (Manouselis et al, 2011).

One important example of TEL in the context of programming education is Scratch (Malan and Leitner, 2007), a visual programming language developed and maintained by the Massachusetts Institute of Technology (MIT). Thanks to its intuitive style, Scratch significantly reduces the learning curve in children and beginners in computer programming. The key educational approach promoted by Scratch is informal learning (European Centre for the Development of Vocational Training, 2014). Basically, this is because students are free to create, explore and reuse programming projects without the teachers presence. In this way, Scratch can also be used for supporting the formal, or traditional, teaching of computer programming in educational environments.

There are some examples of the use of Scratch in higher education (e.g., college) in introductory Computer Science courses with Scratch, this language successfully familiarized inexperienced students with the fundamentals of programming (Malan and Leitner, 2007). Similarly, Scratch has been shown to help students understand programming by offering a different perspective from traditional environments (Wolz et al, 2008). The same authors reported that the transition of the students to Java or C programming languages becomes easier after learning Scratch (Wolz et al, 2009).

Since there are benefits to merging formal and informal learning, we have included Scratch as an educational resource for teaching the subject of Foundations of Computer Programming at Universidad Estatal de Milagro (UNEMI) (Ecuador). Since this started in 2015, there has been an increased motivation and academic performance among the students. Nevertheless, some issues pertaining to the usability of Scratch were also noticed: (i) there is a gap between the employed learning approaches since professors are constrained to classroom activities, and (ii) certain students feel unmotivated because they are confronted with programming exercises that do not fulfill their individual expectations (e.g., exercises are too easy/too complex). This latter issue has been experienced previously (Tanrikulu and Schaefer, 2011).

Thus, the question arises: How to improve learning with Scratch, while saving the gap between formal and informal learning approaches, and allow students to do exercises according to their levels of knowledge and previous experiences? An intuitive and effective solution to this problem comes from Recommender Systems (RSs) (Herlocker et al, 2004) (Ricci et al, 2011). Literature reflects several examples of significant benefits of RSs in educational environments (e.g., reviewed in (Manouselis et al, 2013) and (Klašnja-Milićević et al, 2015)).

Bearing in mind the above motivations, in this chapter we propose an integrated approach to improving the learning process with Scratch for college students. This

approach consists of a simple Web Application that includes Scratch as a project editor along with an RS for exercises (i.e., problem statements). These exercises are created by the professor and included in the system. Thus, the student can access the exercises, solve them with Scratch, and evaluate them according to two criteria: appeal and complexity.

Based on the record of evaluated exercises, the system recommends new exercises to the student via a collaborative filtering approach. The assumption is that students with similar evaluations over a given set of exercises are a good source for recommendation. From a pedagogical perspective, this proposal can be considered a mediator between formal and informal learning approaches. In other words, it not only fills the gap between professors lectures and student interaction with Scratch, it also helps to develop autonomous learning.

The rest of the chapter goes as follows: Section 2 provides a brief overview of Scratch. Next, we review related works about Recommender systems in learning environments (Section 3). In Section 4 the proposed approach is presented, being some preliminary discussed in Section 5. Finally, the conclusion and future works are outlined in Section 6.

2 What is *Scratch*?

Scratch is a free visual programming language (VLP) created by the MIT Media Lab Lifelong Kindergarten Group in 2005. A VLP maps components of high-level programming languages into equivalent visual concepts (Jost et al, 2014). So, VLPs can significantly reduce the learning effort in beginners of computer programming (e.g. children). This is the case of Scratch.

Currently, Scratch goes beyond a simple VLP, it also involves an online community and other websites devoted to share projects and experiences. In Figure 1 the home page of the Scratch website is shown. One important example of these websites extending Scratch is *ScratchED* [1]. It is developed and supported by the Harvard Graduate School of Education.

ScratchEd involves an online community where Scratch educators share stories, exchange resources, ask questions, and find people. This project organizes the educational resources by means of categories, such as: Education level, Context type, Curricular area and Language. For example, the Education level category incorporates resources for all educational levels, that is, from the Preschool and Kindergarten to College and University. Particularly, in the last level more of 326 resources exist for enhancing the autonomous learning in college students.

In order to illustrate how certain computer science concepts can be mapped into Scratch's visual components consider Figure 2. It corresponds to a fragment of a resource named *Computational Concepts Supported in Scratch* appearing in the ScratchEd website. Notice that since the resource is in tabular form it is very self-

[1] http://scratched.gse.harvard.edu/

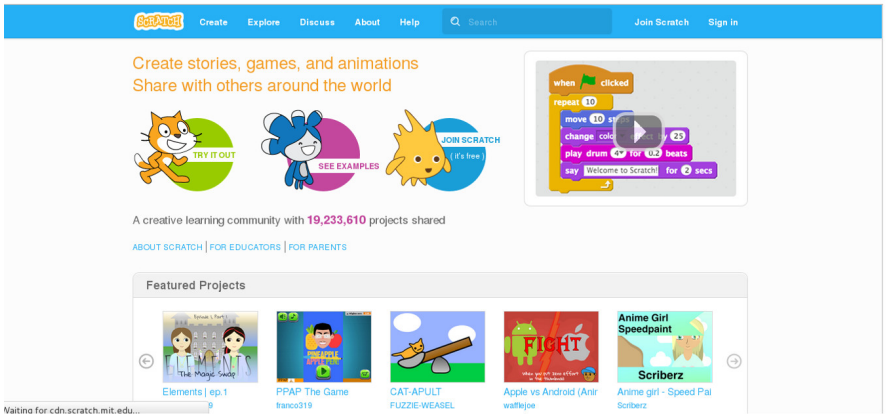


Fig. 1 Scratch Environment.

explanatory. For sake of simplicity consider the third row corresponding to the *random* concept. As stated in column *Explanation*, using the *pick random* component a random integer number can be generated within a given range. The corresponding example depicted in the column *Example* gives a clear idea of how to use such a component in the Scratch language.

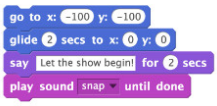
Concept	Explanation	Example
sequence	To create a program in Scratch, you need to think systematically about the order of steps.	
iteration (looping)	*forever* and *repeat* can be used for iteration (repeating a series of instructions)	
random	*pick random* selects random integers within a given range.	
conditional statements	*if* and *if else* check for a condition.	
boolean logic	*and*, *or*, *not* are examples of boolean logic	

Fig. 2 Mapping of Computer Science concepts into Scratch's components.

3 Recommender systems and learning environments

An RS is a software tool that seeks to determine, and suggests, what a particular user will find useful (Ricci et al, 2011). It is also considered to be part of the so-called information filtering system, which exploit the user information for predicting ratings or preferences that the user would give to an actual item (Bobadilla et al, 2013; Lu et al, 2015). 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 (Ricci et al, 2011) five types of RSs exists: content-based, knowledge-based, demographic, community-based, collaborative and hybrid. Collaborative Filtering RSs have been perhaps, the type most widely used (Desrosiers and Karypis, 2011; Elahi et al, 2016). It is built on the assumption that one user may like items that other users with similar tastes liked in the past. This is more or less the assumption adopted for the RS that was implemented in the present work. Specially, CFRS involves two major approaches: user-user and item-item (Elahi et al, 2016). In general, both approaches rely on the Nearest Neighbors algorithm (Desrosiers and Karypis, 2011).

One of the major areas where RSs have been broadly applied in e-learning environments. That is, within the context of TEL (Manouselis et al, 2011) to improve the 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, courses) to educational participants (e.g., students and/or teachers).

Related literature shows a wide variety of works proposing RSs for e-learning environments. Some of the most important works in the field are briefly explained next. For an in-depth survey, the reader is referred to (Manouselis et al, 2013) and more recently to (Klašnja-Milićević et al, 2015).

The RS *CourseAgent* was conceived with the idea of enabling students to provide feedback in implicit and explicit ways (Farzan and Brusilovsky, 2006). This RS allows the students to directly evaluate courses with respect to (i) their relevance regarding each career goal and (ii) the level of difficulty of the course. Both parameters provide implicit feedback when students plan or register in a course. The basic and evident benefits of this RS for the students are that (i) it constitutes a course management system that retains the information about the classes taken, and (ii) it facilitates communication with student advisors.

The Virtual University of Tunis develops automatic recommendations in e-learning platforms (Khribi et al, 2009). They 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 student needs and goals, and then predict a recommendation list.

It has been argued that traditional RSs are not suitable for supporting e-learning because, up to now, they have not taken into account two important mechanisms: the learning processes and the analysis of social interaction (Wan and Okamoto, 2011). To deal with these issues, the authors of the argument proposed a flexible approach involving a multidimensional recommendation model and a Markov Chain Model.

The results showed that more suitable recommendations can be obtained from that approach. Similar research with a personalized approach was proposed. It relied on data mining and natural language process technologies for determining learner relationships based on learning processes and learning activities (Wan et al, 2011).

In order to guide learners in personalized, inclusive e-learning scenarios, an important analysis was conducted on how RSs can be applied to e-learning systems (Santos and Boticario, 2011). In that study, three technological requirements for developing semantic education RSs were provided. Other authors have reported successful experience using similar ideas. See for example (Ghauth and Abdullah, 2011; Lee et al, 2012; Dwivedi and Bharadwaj, 2013; Tewari et al, 2015).

A framework for rapid prototyping of knowledge-based RSs was also reported in (Ruiz-Iniesta et al, 2012), and was used for recommending learning objects. From a software development perspective, the proposed framework is flexible enough for implementing new approaches since it includes default implementations of alternative strategies for each of its five stages. Two RSs were implemented in order to illustrate the benefits of this framework.

Related to the technological benefits of RS in education, there are subjective dimensions in this topic which are also important to study. For instance, a psychological view to learning with personalized RSs was provided by (Buder and Schwind, 2012). Here, a very good fit between the main features of RSs (collective responsibility, collective intelligence, user control, guidance, personalization) and the principles in learning sciences is demonstrated. However, the authors claim 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, some 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 (Peiris and Gallupe, 2012) a conceptual framework is proposed for explaining how evolving recommender-driven online learning systems (ROLS) support students, course authors, course instructors, system administrators and policy makers in development. Moreover, this framework involves 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 in which an influence diagram included the observable variables for assessing the collaboration among users (Anaya et al, 2013). By applying machine learning techniques, the influence diagram was refined to increase its accuracy. The main outcome of this work was the development of an automatic RS together with a pedagogical support system in the form of a decision tree, which provides visual explanation to the user.

A generic cloud-based architecture for a system that recommends learning elements according to the affective state of the learner was also presented (Leony et al, 2013). The authors also provided some use cases, explaining the implementation of one of them. 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 (Erdt et al, 2015). From an in-depth survey obtained from 235 works on the subject, it was concluded that there is an important interest in designing better strategies to evaluate 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 an increasing number of studies;
2. most of the existing works are technology-based, that is, they proposed RSs for enhancing the learning process. In contrast, just a few works employ a subjective perspective like (Buder and Schwind, 2012) for analyzing the role of RS in this context; and
3. up to our knowledge there are not RS supporting programming learning with Scratch.

Conclusions 1 and 3 gave us additional reasons for proposing the system explained next.

4 Scratch+ERS: an integrated approach

The system proposed in this chapter aims to improve the current state of teaching Foundations of Computer Programming at Universidad Estatal de Milagro (State University of Milagro), Ecuador, with Scratch. Figure 3 helps demonstrate both the current and the proposed approaches for such a process.

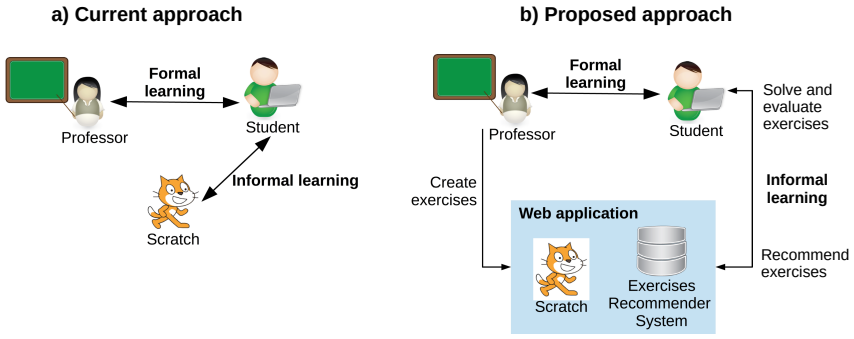


Fig. 3 Comparison between the current and the proposed approaches.

In the first approach (Fig.3-a), the professor interacts with students exclusively by means of classroom lectures (formal learning). The students use Scratch in the tra-

ditional form, that is, by creating or modifying projects (informal learning). Despite the benefits of this method, the professor has difficulty 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. In addition to this issue, as we observed over the course of one year, certain students were not satisfied with the complexity of the exercises they are asked to solve with Scratch. For instance, the more experienced students are faced with too-easy Scratch projects.

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 an individual characterization of the students in order 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 student characterization and the suggested exercises are expected to evolve over time. Hence, this assignation process will be repeated again and again.

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

With that in mind, a modification is therefore depicted (see Figure 3-b)): it's a simple Web Application composed of Scratch as a project editor along with an RS of exercises. Notice that this proposal not only allows the interactions of the current approach, but also personalizes the learning process of students using Scratch. In this way, the professor is able to control students learning process 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, under the assumption that students with similar tastes and complexity perceptions about exercises are a good source for recommendations.

From a pedagogical perspective, we consider this proposal to be a new mediator between professors and students. In the following sections, the proposal is presented in detail. First, the developed Web Application is described through its main modules and features. Then, the technical aspects of the implemented RS are explained.

4.1 Web Application

The Web Application has an easy-to-use graphic user interface (GUI), and was developed using PHP and MySQL server as a database system.

In Figure 4, you can see that in the central zone of the applications main window, Scratch is embedded as an editor, while in the upper and right zones the rest of the added modules appear. In the upper zone, the user area provides functionalities in terms of user access to the system. The user interface for the exercise RS appears in the right area of the screen. This module, from top to bottom, consists of:

- a clock for measuring how much time the student spent on the exercise
- the exercise statement
- a pool of recommended exercises
- a pool of all available exercises
- a system for evaluating the taste and complexity of the active exercise.

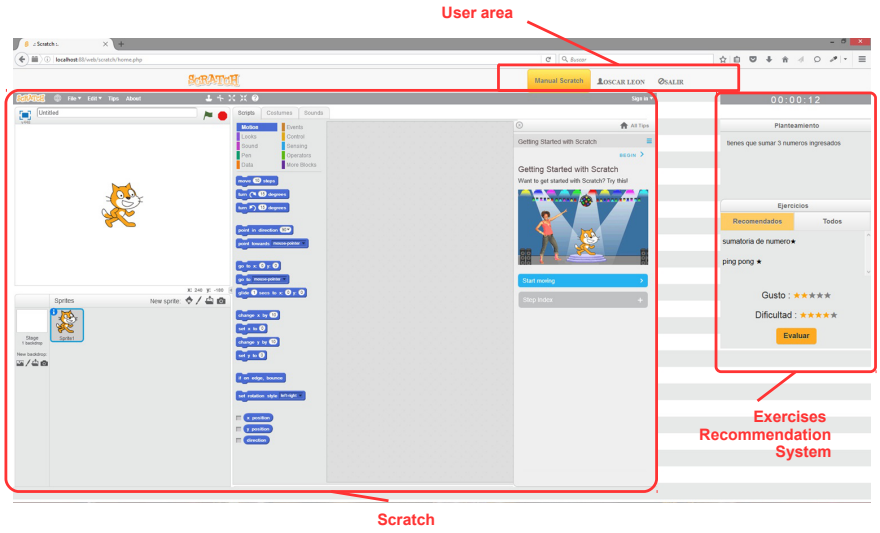


Fig. 4 Main screen of the proposed Web Application.

The main workflow of the interaction between the student and the proposed system is also depicted in Figure 5. You can see that, after login, the system checks whether the student has previously seen an exercise. If the student has, then 10 exercises are recommended following a collaborative filtering approach. Otherwise, the system selects the easiest and most tasteful exercises among all available exercises. Regardless, the system builds a list of 10 exercises that are presented to the student.

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

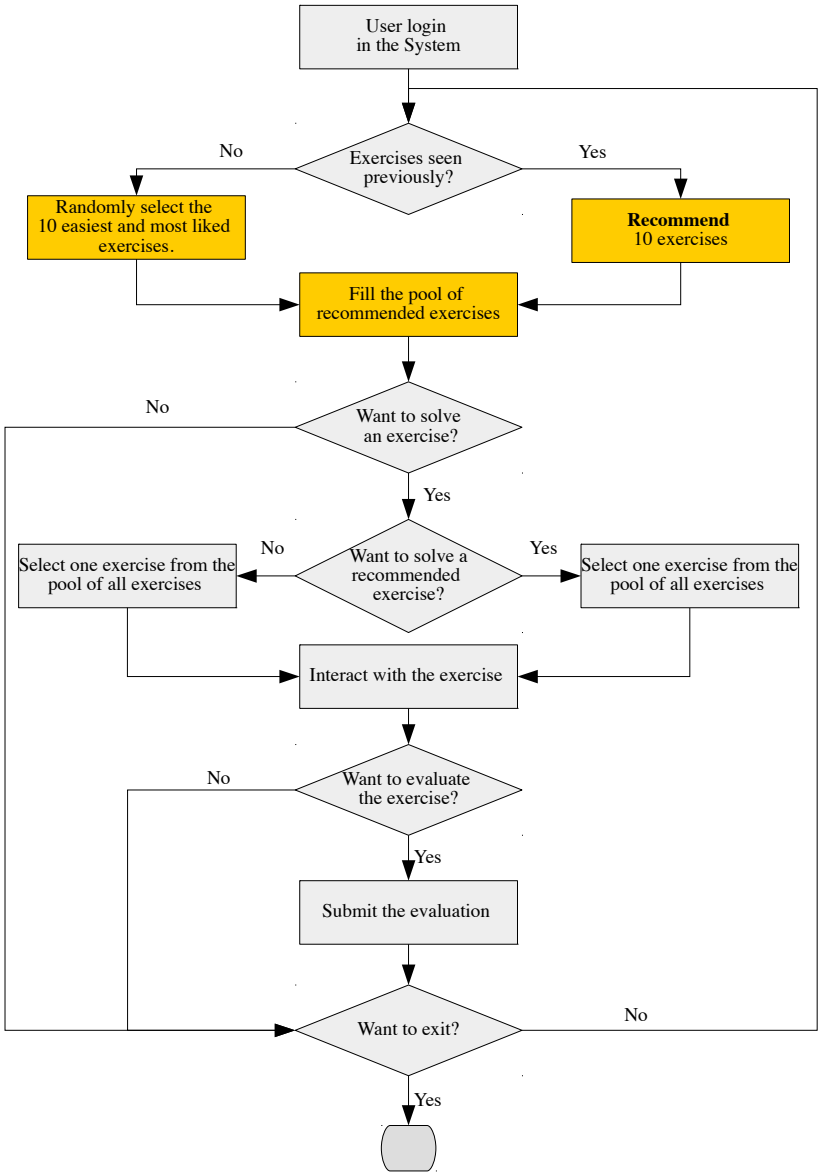


Fig. 5 Main flow of the proposed Web Application.

4.2 Technical aspects of the recommender system

The RS included in our Web Application is based on a collaborative filtering approach, using both users and items (Bobadilla et al, 2013; Lu et al, 2015). Evidently, in this context, the users are the students, while the items are the exercises to solve in Scratch. The aim of this RS is to exploit the experience of existing students in order to suggest suitable exercises to a certain student. In what follows we refer such a student as *active student*.

Formally, 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* ($I \in \{1,2,3,4,5\}$). 1) 1) It 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* ($C \in \{1,2,3\}$). It allows stating the complexity level of the exercise from the student point of view. For this variable, 1 means a low complexity, 2 a medium complexity, and 3 a high complexity.

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

4.2.1 Cold start

During the implementation of an RS in real environments a common difficult is how to recommends when no user experience or data exist. This issue is known as *Cold start* (Herlocker et al, 2004). In the case of our RS, we have two scenarios:

1. *the system hasn't any 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
2. *the active student hasn't any experience recorded by the system*. This is the case of new users. Here, the system recommends exercises that are most frequently evaluated by the community with $I = 3$ and $C = 1$. This assures that new users start with the most popular and easiest exercises.

4.2.2 Neighborhood computation

A user-user collaborative filtering approach is adopted here. As mentioned before, the first step consists of finding the students who are most similar with respect to

the active user k. To this end, the Cosine similarity function between two vectors (Bramer, 2013) was 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 applied to each evaluation criteria independently.

More formally, $V \in \{I, C\}$ is an evaluation criteria, and $\mathbf{v}_k$ and $\mathbf{v}_i$ the evaluation vectors for the exercises that the active student u_k and the student u_i have in common, respectively. Thus, the similarity between students u_k and u_i is computed as:

$$S_V(u_k, u_i) = \frac{\mathbf{v}_k \cdot \mathbf{v}_i}{\|\mathbf{v}_k\| \cdot \|\mathbf{v}_i\|} \tag{1}$$

S_V takes 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 Eq. 1 is not able to express the significance of the number of common exercises with respect to the record of all exercises. In other words, since Eq. 1 computes similarity by using information from the exercises that both students have in common, it does not take into account their corresponding records of exercises. We argue that this information is crucial for obtaining a suitable computing of similarity between two given students. In this regard, the following expression was employed:

$$S_P(u_k, u_i) = \frac{|H_k \cap H_i|}{|H_k|} \tag{2}$$

here, 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 Eq. 2 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 defined also in the range $[0, 1]$, with possible values the same meaning as Eq. 1.

So we have three sources for computing similarity for every pair of students: S_i, S_v, and S_p. The question is how to aggregate them in order to obtain a single value for the 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:

$$S(u_k, u_i) = S_I(u_k, u_i) \cdot S_C(u_k, u_i) \cdot S_P(u_k, u_i) \tag{3}$$

here, S_I and S_C are defined by Eq. 1, while S_P is given by Eq. 2. Notice that since S_I, S_C and S_P take values in $[0, 1]$, then S will also take values in this range.

The final step after computing similarity with Eq. 3 is to sort the students according to their corresponding similarity values regarding the active user k. This sorting operation is performed in descending order involving students with $S > 0$. The obtained set of sorted students is denoted by U_k.

4.2.3 Building the list of recommended exercises

In this process, a list of exercises recommended for the active user u_k is created by iteration through U_k. Specifically, for each student of the U_k set, exercises not evaluated by the active user are added to the list. This process runs until the list is completed with 10 exercises, or no more students from U_k remain to be analyzed. In this latter case, the list is completed with the exercises that the community more frequently evaluates, but not previously evaluated by the active user.

5 Preliminary results

Evaluating RSs in educational environments is an ongoing research topic. It is indeed a complex process because it involves several goals that are hard to assess (e.g., knowledge acquired by the student and user satisfaction, among others). These evaluation goals can be grouped into three broad categories: *RS performance*, *user-centric effects* and *learning effects* (Erdt et al, 2015).

Due to the characteristics of our educational environment and the relative short time of use of the proposed system, a *Real-Life Testing* methodology was adopted (Erdt et al, 2015). Its main goal is to assess the system according to perceived satisfaction of real users (i.e., the students).

A total of 64 students from both Computer Science and Industrial Engineering participated in the analysis. Both of these areas of study include the Fundamentals of Computer Programming course in their curricula. However, more motivation was expected of Computer Science students than of the Industrial Engineering students. Such a difference can be useful for assessing the system from two distinct viewpoints.

After using the system over a period of three months, we gave the students a questionnaire asking their opinion of nine assertions. Table I shows these assertions according to the above-mentioned three evaluation goals (Erdt et al, 2015). More emphasis was put on evaluating both learning and user-centric effects (e.g., more assertions are included for assessing these goals). In this context, they seem to be more valuable criteria than the RS performance since:

1. it is difficult to assess accuracy in the presented RS if no data are available for conducting offline experiments; and
2. the system will take no more than 100 users; consequently, when users are accessing the Web Application, it's expected there will be no system overload. In addition, for that number of users the similarity computations and the sorting operation explained in the previous section can be performed in an efficient manner.

However, 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 offline experiment, since they were measured based

on opinions of real users. However, the authors are aware that a greater number of technical tests will be needed, which will be the subject of future works.

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

Table 1 List of assertions employed in the questionnaires for assessing the proposed system.

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

The results of the questionnaires (Figure 6) were organized into three groups: (a) Satisfaction of Computer Science students, (b) Satisfaction of Industrial Engineering students and (c) Overall satisfaction. The latter is the aggregation of the first two groups. A generally acceptable degree of satisfaction is appreciated. For example, more than 50% of the students at least agreed with all the assertions.

However, differences exist between the first two groups. As expected, Computer Science students are more critical than Industrial Engineering students (Figure 6-a and Figure 6-b). In both cases, a significant number of students do not agree with A1, indicating that much more work has to be done regarding the systems 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 careers 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 6-c).

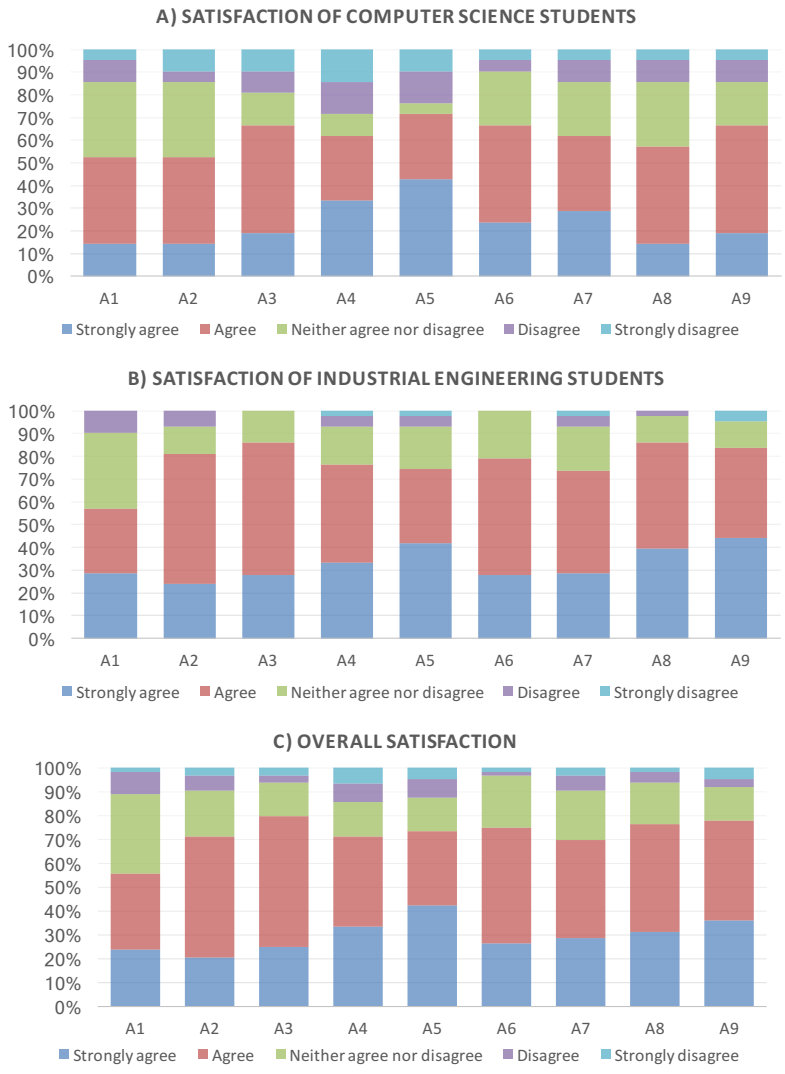


Fig. 6 Satisfaction of students with the system according to the questions.

6 Conclusion and future work

In this chapter, an integrated approach for improving computer programming students learning process with Scratch is proposed. Our previous experience regarding the use of Scratch for complementing the teaching process of Foundation of Computer Programming at Universidad Estatal de Milagro, Ecuador, provides important evidence that such a learning strategy can be improved. Specifically, for solving two issues: (1) the current gap between the formal and informal learning approaches and (2) a personalized interaction between the student and Scratch.

An easy-to-use Web Application was therefore developed that involves Scratch along with a recommender system for exercises. A Real-Life Testing methodology was adopted in order to validate this approach. 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.

However, this is considered to be a first step towards having a better system. Future works will be oriented to improve both, the recommender system performance and Web Application usability. Besides, we will explore the inclusion of our proposal into the Scratch source code with aims of sharing its benefits with the existing community.

Acknowledgements This research has been conducted during the development of the project FOCICYT: *Soft Computing Applications in Higher Education Environments*, which is financed by the Technical State University of Quevedo.

References

Anaya AR, Luque M, García-Saiz T (2013) Recommender system in collaborative learning environment using an influence diagram. Expert Systems with Applications 40(18):7193–7202

Bobadilla J, Ortega F, Hernando A, Gutierrez A (2013) Recommender systems survey. Knowledge-Based Systems 46:109–132

Bramer M (2013) Introduction to Data Mining, Springer London, London, pp 1–8

Buder J, Schwind C (2012) Learning with personalized recommender systems: A psychological view. Computers in Human Behavior 28(1):207–216

Dascalu MI, Bodea CN, Moldoveanu A, Mohora A, Lytras M, de Pablos PO (2015) A recommender agent based on learning styles for better virtual collaborative learning experiences. Computers in Human Behavior 45:243 – 253

Desrosiers C, Karypis G (2011) A Comprehensive Survey of Neighborhood-based Recommendation Methods, Springer US, chap Recommender Systems Handbook, pp 107–144

Do P, Le H, Nguyen VT, Dung TN (2014) StudyAdvisor: A Context-Aware Recommendation Application in e-Learning Environment, Springer International Publishing, Cham, pp 119–128

Dwivedi P, Bharadwaj KK (2013) Effective trust-aware E-learning recommender system based on learning styles and knowledge levels. Educational Technology and Society 16(4):201–216

Elahi M, Ricci F, Rubens N (2016) A survey of active learning in collaborative filtering recommender systems. Computer Science Review 20:29–50

Erdt M, Fernández A, Rensing C (2015) Evaluating Recommender Systems for Technology Enhanced Learning: A Quantitative Survey. IEEE Transactions on Learning Technologies 8(4):326–344

European Centre for the Development of Vocational Training (2014) Terminology of European education and training policy: a selection of 130 key terms, 2nd edn. Luxembourg: Publications Office of the European Union

Farzan R, Brusilovsky P (2006) Social Navigation Support in a Course Recommendation System, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 91–100

Ghauth K, Abdullah N (2011) The effect of incorporating good learners' ratings in e-learning contentbased recommender system. Educational Technology and Society 14(2):248–257

Gomes LFAM, Colcher R, Wolcott P, Herrera-Viedma E, Shi Y, Tejeda-Lorente A, Bernabe-Moreno J, Porcel C, Galindo-Moreno P, Herrera-Viedma E (2015) 3rd international conference on information technology and quantitative management, itqm 2015 a dynamic recommender system as reinforcement for personalized education by a fuzzly linguistic web system. Procedia Computer Science 55:1143 – 1150

Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53

Jost B, Ketterl M, Budde R, Leimbach T (2014) Graphical programming environments for educational robots: Open roberta - yet another one? In: 2014 IEEE International Symposium on Multimedia, pp 381–386

Khribi M, Jemni M, Nasraoui O (2009) Automatic recommendations for e-learning personalization based on web usage mining techniques and information retrieval. Educational Technology & Society 12(4):30–42

Klašnja-Milićević A, Ivanović M, Nanopoulos A (2015) Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions. Artificial Intelligence Review 44(4):571–604

Lee G, Salehi M, Kmalabadi IN (2012) 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 2:565–570

Leony D, Parada Gélvez H, Mnoz-Merino P, Pardo A, Kloos C (2013) A generic architecture for emotion-based recommender systems in cloud learning environments. Journal of Universal Computer Science 19(14):2075–2092

Lu J, Wu D, Mao M, Wang W, Zhang G (2015) Recommender system application developments: A survey. Decision Support Systems 74:12 – 32

Malan DJ, Leitner HH (2007) Scratch for budding computer scientists. In: Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, ACM, New York, NY, USA, SIGCSE '07, pp 223–227

Manouselis N, Drachsler H, Vuorikari R, Hummel H, Koper R (2011) Recommender Systems in Technology Enhanced Learning, Springer US, Boston, MA, pp 387–415

Manouselis N, Drachsler H, Verbert K, Duval E (2013) Survey and Analysis of TEL Recommender Systems, Springer New York, New York, NY, pp 37–61

Peiris K, Gallupe Rc (2012) A Conceptual Framework for Evolving, Recommender Online Learning Systems. Decision Sciences Journal of Innovative Education 10(3):389–412

Ravichandran M, Kulanthaivel G (2014) Intelligent multi agent based hybrid recommender model in E-learning system. International Journal of Applied Engineering Research 9(24):29,949–29,960

Ricci F, Rokach L, Shapira B (2011) Recommender Systems Handbook, Springer US, Boston, MA, chap Introduction to Recommender Systems Handbook, pp 1–35

Ruiz-Iniesta A, Jiménez-Díaz G, Gómez-Albarrán M (2012) A framework for the rapid prototyping of knowledge-based recommender systems in the learning domain. Journal of Research and Practice in Information Technology 44(2):167–181

Santos OC, Boticario JG (2011) Requirements for semantic educational recommender systems in formal e-learning scenarios. Algorithms 4(2):131–154

Shishehchi S, Banihashem SY, Mat Zin NA, Noah SAM (2012) Ontological approach in knowledge based recommender system to develop the quality of e-learning system. Australian Journal of Basic and Applied Sciences 6(2):115–123

Tanrikulu E, Schaefer BC (2011) The users who touched the ceiling of scratch. In: Yalin H, Adiloglu F, Boz H, Karata S, and FO (eds) World Conference on Educational Technology Researches - 2011, vol 28, pp 764–769

Tewari AS, Saroj A, Barman AG (2015) E-learning recommender system for teachers using opinion mining. Lecture Notes in Electrical Engineering 339:1021–1029

Vesin B, Klašnja-Milićević A, Ivanović M, Budimac Z (2013) Applying recommender systems and adaptive hypermedia for e-learning personalization. Computing and Informatics 32(3):629–659

Wan X, Okamoto T (2011) Utilizing learning process to improve recommender system for group learning support. Neural Computing and Applications 20(5):611–621

Wan X, Jamaliding Q, Okamoto T (2011) Analyzing learners' relationship to improve the quality of recommender system for group learning support. Journal of Computers 6(2):254–262

Wolz U, Maloney J, Pulimood SM (2008) scratch your way to introductory cs. In: Covington K (ed) 39th SIGCSE Technical Symposium on Computer Science Education, vol 40, pp 298–299

Wolz U, Leitner HH, Malan DJ, Malone (2009) Starting with scratch in cs1. In: Covington K (ed) 40th SIGCSE Technical Symposium on Computer Science Education, vol 41, pp 2–3