Enhancing the learning of programming using Scratch

A recommender-systems-based approach in non WEIRD communities

Jesennia del Pilar Cárdenas Cobo

Advised by:
Dr. David Benavides Cuevas and Dr. José A. Galindo

First published in April 2020 by
Departamento de lenguajes y sistemas informáticos, ETSI Informática
Avda. de la Reina Mercedes s/n
Sevilla, 41012. SPAIN

Classification (ACM 2019):

Categories and subject descriptors:

[500]Applied computing-Computer-managed instruction
[500]Software and its engineering-Empirical software validation
[500]Applied computing-Collaborative learning
[500]Applied computing-Learning management systems
[500]Applied computing-Computer-assisted instruction

General Terms: Design, Theory, Algorithms, Computational thinking.

Additional Key Words and Phrases: Fundamentals of programming and programming language, academic performance, Scratch, recommender systems, visual programming languages, programming learning and educational programming learning.

Dr. David Benavides Cuevas y Dr. José A. Galindo, de la Universidad de Sevilla.

HACEN CONSTAR

que Jesennia del Pilar Cárdenas Cobo, ha realizado bajo nuestra supervisión el trabajo de investigación titulado

Enhancing the learning of programming using Scratch, a recommender-systems-based approach in non WEIRD communities

Una vez revisado, autorizamos el comienzo de los trámites para su presentación como tesis doctoral al tribunal que ha de juzgarlo.

Fdo. Dr. David Benavides Cuevas y Dr. José A. Galindo
Universidad de Sevilla,
Sevilla, febrero de 2020

Yo, Jesennia del Pilar Cárdenas Cobo, con número de DNI 0918224783,

DECLARO

Ser la autora del trabajo que se presenta en la memoria de esta tesis doctoral que tiene por título:

Enhancing the learning of programming using Scratch, a recommender-systems-based approach in non WEIRD communities

Lo cual firmo en Sevilla, febrero de 2020.

Fdo. Jesennia del Pilar Cárdenas Cobo

A mi familia:
Teresa, Olga, Segundo, Juan Carlos, Carlos y Ámbar.

Contents

List of Figures

List of Tables

Acknowledgements

Expresar todo el agradecimiento que siento en unas pocas líneas es difícil, sin embargo quiero iniciar por agradecer a Dios por mi existencia y por todas las bendiciones recibidas.

Gracias a mis padres y abuelita, que con su apoyo incondicional me alentaron cada día para seguir; a mi esposo e hijos por comprender mi ausencia; a mi Director Dr. David Benavides Cuevas, sin su guía hubiese sido difícil poder avanzar; a mi Director Dr. José A. Galindo, punto de apoyo valioso para culminar este trabajo.

Gracias a la Universidad Estatal de Milagro y a sus autoridades, por confiar en nuestro trabajo de investigación y permitir que bajo su insignia propaguemos el anhelo de lograr que el pensamiento computacional se desarrolle en nuestra región. A la Universidad de Sevilla por acojerme en su programa doctoral y darme la oportunidad de formarme. Al Dr. Miguel Toro, pues bastó solo una ocasión de escuchar sus ideas, para orientar mi trabajo.

Gracias al Dr. Jesús Moreno, pues su experiencia y trabajo realizado, me ha servido para enfocar los objetivos alcanzados. Al Dr. Marcos Román, por permitirme aplicar sus resultados en la evaluación de nuestra intervención en los contextos escolares.

Gracias a mis amigos Dr. Amilkar Puris y Dr. Pavel Novoa por compartir su experiencia conmigo, a mis pupilos Oscar y Nestor, a los estudiantes, profesores y amigos, quienes permitieron que trabaje con ellos y finalmente pero no menos importante, decirte gracias Hans, tu apoyo ha sido increíble, mantener la mente abierta fue la clave, tenías razón.

Abstract

In today's world, there is a growing need for professionals with computer skills in general, and programming in particular. This is so, both in WEIRD contexts and in contexts that are not. WEIRD is an acronym from English (Western, Educated, Industrialized, Rich and Democratic Societies) and is used to refer to certain sectors of societies that are "Western, educated, industrialized, rich and democratic.

The Ecuadorian State makes a high investment in the training of engineering career professionals offered by public universities. In Ecuador, these careers are highly demanded. However, a high student dropout is verified because of basic courses such as Fundamentals of programming in the first levels, with the consequent deficiencies in the performance of the careers. Additionally, the low qualification of students in computer skills indicates that they have not developed their computational thinking skills. In general terms, this situation contributes to the loss of competitiveness of non-WEIRD countries.

For this reason, a pedagogical tool was introduced in teaching that allowed motivating programming learning, reducing dropout and improving academic performance in introductory programming courses at the university level.

We present empirical evidence of the positive impact of Scratch on the Fundamentals of Programming courses. The use of this pedagogical tool allowed students to develop the concepts of programming logic and the use of basic control structures. Students who used Scratch scored better, reduced the number of suspensions and dropouts, and students were encouraged to enjoy the subject. On the other hand, when developing the exercises with Scratch in combination with the CARAMBA recommendation system, students were motivated to autonomous learning.

The use of CARAMBA showed a positive correlation with the chances of success in the course. Specifically, $>$ 52% success, whose result is 8% higher than the use of Scratch alone (without recommendations) and 21% higher than

traditional education (without Scratch). The Scratch + CARAMBA tools were scaled to a school environment in non-WEIR contexts for programming learning. The post-application evaluation of the instrument indicated increases in exam scores in all grades analyzed. It should be noted that there was an average increase of 32% in the afternoon sections compared to the morning sections.

This work opens a line of future research by bringing a pedagogical tool to different educational environments. The results allow us to propose improvements in CARAMBA's recommendations, especially regarding the variables of usability, interactivity, language and pedagogical aspects. CARAMBA functionality should incorporate educational data mining tools that allow learning models based on the profile of the students. Another aspect that we intend to address is the scalability of the system in order to adapt it to other study scenarios with more users and number of exercises.

Resumen

En el mundo actual, existe cada vez mayor necesidad de profesionales con conocimientos de computación en general, y programación en particular. Esto es así, tanto en contextos WEIRD como en contextos que no lo son. WEIRD es un acrónimo procedente del inglés (Western, Educated, Industrialized, Rich and Democratic Societies) y es utilizado para referirse a ciertos sectores de sociedades que son "occidentales, educadas, industrializadas, ricas y democráticas".

El Estado ecuatoriano realiza una alta inversión en la formación de profesionales de carreras de ingeniería ofertadas por las universidades públicas. En el Ecuador estas carreras son altamente demandadas. Sin embargo, se verifica una alta deserción estudiantil a causa de cursos básicos como Fundamentos de programación en los primeros niveles, con las consecuentes deficiencias en el desempeño de las carreras. Adicionalmente, la baja calificación de los estudiantes en competencias de computación indica que no tengan desarrolladas sus habilidades de pensamiento computacional. En términos generales, esta situación contribuye a la pérdida de la competitividad de los países non WEIRD.

Por esta razón se introdujo en la enseñanza una herramienta pedagógica que permitió motivar el aprendizaje de programación, disminuir la deserción y mejorar el rendimiento académico en los cursos introductorios de programación a nivel universitario.

Presentamos evidencia empírica del impacto positivo de Scratch en los cursos de Fundamentos de Programación. El uso de esta herramienta pedagógica permitió desarrollar en los estudiantes los conceptos de lógica de programación y el uso de estructuras básicas de control. Los estudiantes que usaron Scratch obtuvieron mejores notas, redujeron la cantidad de suspensos y de deserciones, y se alentaron a los estudiantes a disfrutar de la asignatura. Por otro lado, al desarrollar los ejercicios con Scratch en combinación con el sistema de recomendaciones CARAMBA los estudiantes se vieron motivados al aprendizaje autónomo.

El uso de CARAMBA mostró una correlación positiva con las posibilidades de éxito en el curso. Concretamente, ¿52% de éxito, cuyo resultado es 8% superior al uso solo de Scratch (sin recomendaciones) y 21% superior a la enseñanza tradicional (sin Scratch). Las herramientas Scratch+CARAMBA fueron escaladas a un ambiente escolar en contextos non WEIR para el aprendizaje de programación. La evaluación posterior a la aplicación del instrumento indicó incrementos en las notas de los exámenes en todos los grados analizados. Vale destacar, que hubo un incremento del promedio de las notas del 32% en las secciones de la tarde respecto a las secciones matutinas.

Este trabajo abre una línea de investigación futura al llevar una herramienta pedagógica a diferentes entornos educativos. Se pretende plantear mejoras en las recomendaciones de CARAMBA, en especial respecto a las variables de usabilidad, interactividad, lenguaje y de aspectos pedagógicos. En la funcionalidad de CARAMBA se deben incorporar herramientas de minería de datos educacionales que permitan modelos de aprendizaje basados en el perfil de los estudiantes. Otro aspecto que pretendemos abordar es la escalabilidad del sistema a fin de adaptarlo a otros escenarios de estudio con más usuarios y cantidad de ejercicios.

Part I

Preface

Chapter 1
Introduction

One never reaches the total truth, nor is one ever totally estranged from it.

Aristotle

In this dissertation, we present our research work developed with university students from contexts with little access to technology where the learning of programming becomes a difficulty to overcome the subject programming fundamentals. For this reason we proposed the use of a pedagogical tool that allowed motivating the development of computer thinking skills and that was later extended to other educational contexts.

In this chapter we give an overview of the contributions, the research method we used and the publications that support the research of the problem presented.

1.1 Overview

In today's society, where most students are considered digital natives, knowing how to program is a necessary skill for future professionals in order to enhance thinking skills for solving problem [41]. Hence, numerous educational institutions have included this skill in their curricula, even at initial levels (i.e. K-12). However, most of the experiences reported in literature regarding this necessary initiative are in so-called WEIRD countries, meaning those with Western, educated, industrialized, rich and democratic populations [33, 40, 69].

In contrast, Latin America countries, and in particular Ecuador, have very different characteristics. The presence of less-developed areas from a socioeconomic point of view, makes the teaching and learning process of programming a real challenge. It is even more so if you consider that students have a weak basis for solving problems in the field of mathematics and physics. According to [10, 56, 60], this can make learning programming a complex task, even in university settings.

The context described above represents the vast majority of Ecuadorian university students. A concrete example is the Universidad Estatal de Milagro (UNEMI) where most of the students come from vulnerable sectors and have integrated baccalaureates rather than specialized ones.

In this way, the number of university students who possess problem-solving skills is low or simply non-existent. It is expected that many of them will fail in the initial semesters, in which programming subjects are taught. In the case of UNEMI, the aforementioned is observed in the degree courses of Computer Systems Engineering and Industrial Engineering. Historical data shows that both courses have high dropout rates in the initial semesters due to the Fundamentals of Programming subject (FP).

To solve this problem, we present a set of contributions in this document that can be summarized as follows (see Figure §1.1):

We used the Scratch visual programming language as a methodological tool in classes (under the teacher's direction) to introduce basic programming concepts such as the use of variables, conditionals, cycles and some concurrence elements. Experience has shown a considerable improvement in learning and an acceptable level of motivation in students, in this experiment 46% of the experimental group passed the mid-year test, while in the control group only 21% made it.

We developed CARAMBA, a platform that integrates the Scratch language with an exercise recommendation system that personalizes students' informal learning. This platform uses a collaborative system to build personalized recommendations from the students' own experience. In all experiments, the use of this tool for autonomous learning complemented the students' individual needs, which helped to improve their academic performance and motivation. Concretely, the success rate achieved by our proposal was more than 52%, which is 8% more than the rate achieved during a previous experience using Scratch only (no recommendation) and 21% more than the historical results of teaching in the traditional way (without Scratch).

We applied a methodological scheme using Scratch and CARAMBA to develop logical thinking in students aged between eight and twelve in a public school. The experience was developed as a knowledge transfer project in which university students were the main protagonists. The development of the computational thinking of school students was validated with the Test-V2 instrument by Román et.al. [68], presenting a growth rate of 30% after the intervention. In addition, the results obtained in the motivation of the university students were encouraging, since only 96% confirm that have been motivated to prepare themselves in the area of programming, prior to the start of the project.

At the time of the intervention with UNEMI students, there were high rates of failure in FP in Computer Systems Engineering and Industrial Engineering. Our thesis presents the results of the didactic implementation of a programming learning tool to support the conceptualization of programming logic and the use of certain basic control structures. After the intervention, better results in academic performance and motivation in the use of the proposed tool were evident.

Furthermore, it was possible to extend the tool to other educational contexts, opening up the opportunity to intervene in diverse social environments where the tool has been validated.

1.2 Research approach

In this research we adopted a quasi-experimental method [70] under an empirical approach to measuring the incidence of the use of technological tools in the learning of university students. The problematical identification of the high rate of retention in the subject FP led to the search for teaching and learning alternatives to help improve this indicator, which fostered a better scenario for students

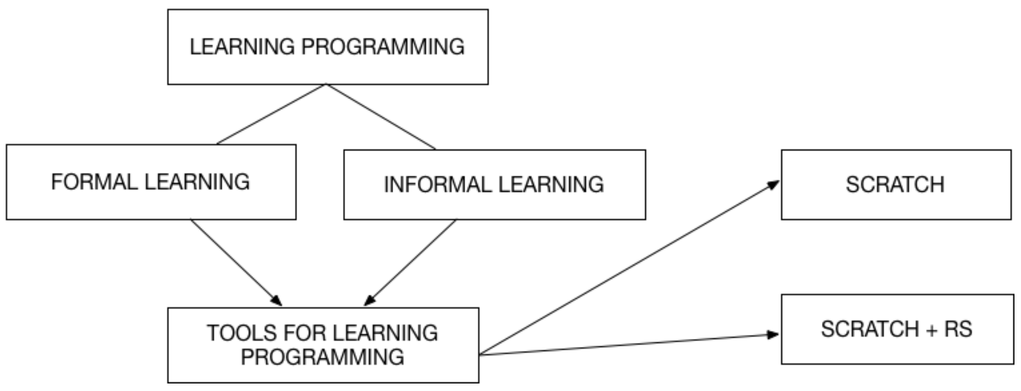


Figure 1.1: Proposal design

In the same way we applied the procedures shown in [37], where the traditional research method is adapted to the nature of technological research. The researcher starts out by collecting the requirements stated by existing users or by new potential users. When the requirements have been collected, the researcher starts making an artifact which is supposed to satisfy the requirements. The overall hypothesis of the technology researcher is: The artifact satisfied the need. The predictions form the starting point of evaluation (hypothesis testing). To the researcher, evaluation may be just as challenging as the manufacturing artifact. If the results are positive, the researcher may argue that the artifact satisfies the need. In case the results are negative, new interactions in the research cycle: problem analysis - innovation - evaluation may be stimulated. (Fig.§1.2)

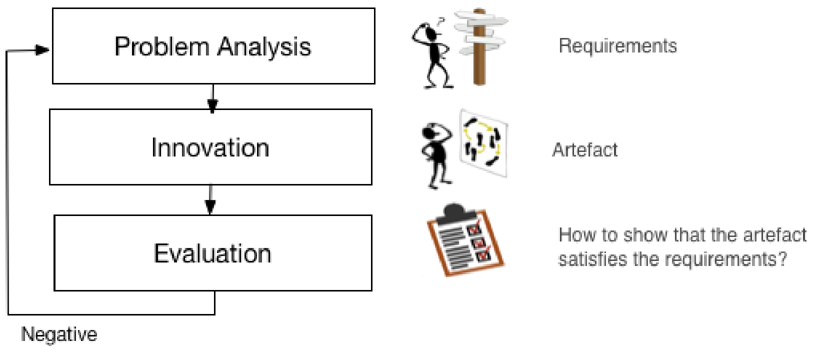


Figure 1.2: Research method

Next, we define the cycle used in each stage of our research in more detail:

Pilot test: The goal of this stage was to determine the cognitive and emotional impact that Scratch has on students as a formal teaching-learning method in the FP course. The methodological specifications were the following:

- Analysis of the problem: The failure rate of students in the subject FP . Why do students fail?
- Innovation: Scratch is introduced in the teaching learning process of the subject using formal learning.
- Evaluation: Improvement in academic performance is obtained, however the increase in motivation was not representative.

Recommender system: The goal was to determine the cognitive and emotional impact of using personalized autonomous learning on students.

- Analysis of the problem: The student failure rate in FP. Why don't students consider Scratch to help them passing the course?
- Innovation: A platform integrating Scratch with an exercise recommendation system to customize autonomous learning is developed. In this way formal and informal learning are integrated into the teaching process.
- Evaluation: Academic achievement is improved and the interest in using the tool is satisfactory

Application: The aim was to adopt previously applied schemes in an educational scenario with different characteristics to measure their impact.

- Analysis of the problem: Low motivation level and poor problem solution ability . Why is there little motivation in school students?
- Innovation: A linkage project is being developed that implements a methodological strategy using Scratch and CARAMBA or learning programming, involving children between eight and twelve years of age.
- Evaluation: Students' motivation in general increases and another phase is planned for the project.

1.2.1 Experimental design

Experiments were carried out in all these three stages to measure the impact of using technological tools on students' learning. Below there is a brief description design of each case.

- Pilot test: For this stage two groups of students (control and experimental), were set, each one applied a different method of teaching and learning (traditional and assisted with Scratch). At the beginning of the experiment, homogeneity tests were carried out on the groups depending on personal, social and economic variables. After a period of intervention, the two groups were given the same knowledge review and the results were compared. Satisfaction analysis was also performed for the group that worked with Scratch.
- Recommendation system: In this case, we made a similar design to the previous one but added a third experimental group that used CARAMBA tool for autonomous learning. Also, we built a pretest-postest experiment where the three groups faced the same test at the beginning and end of the experiment. In this case the group homogeneity test was carried out on the basis of the results achieved in the initial examination. After a period of implementation of the instrument, we carried out a comparative analysis of final results. We also measured the level of motivation of the group that used CARAMBA tool.
- Application: The experimental design in this case was oriented to implement two phases. In the first phase, Scratch was used and after a period CARAMBA was introduced. At the end of both periods the level of motivation of each group of students was analyzed.

To obtain information from the experiments, we defined two research instruments and applied them depending on the measurements to be performed. Both instruments are defined below and identified in the phases of their implementation:

- Knowledge test: This instrument allowed us to measure the cognitive levels achieved by the students until the moment of its application. It was elaborated by a group of teachers of the subject, which guaranteed consistency between questions and acquired knowledge. In addition, whenever the instrument was used, the same test was presented for groups of students involved in the experiment. This instrument was used in the first two stages(Pilot test and Recommendation system).

- Satisfaction questionnaire: To measure satisfaction levels after the application of each experiment, a questionnaire was designed, with questions previously analyzed by a group of experts. This instrument was applied only to the pilot groups in the first two stages and to all groups in the third stage of the investigation.

1.3 Contributions

This section details the sequence in which the results of the thesis have been obtained. Organized in three chapters which detail the methodology, development, evaluation, results and conclusions from the pilot and implementation phase. Figure §1.3 presents the three contributions of this work and then a general description of each of them is made.

1.3.1 Summary of contributions

- Applying Scratch to university students: This contribution was developed with the aim of measuring the impact of using technology as basic programming concepts in the learning process.
- Applying Scratch Extension to students: This contribution was done to measure the impact of a tool for informal and personalized learning on motivation and learning .
- Scratch application and its extension to school children: This research arises from the need to extend the results obtained in previous contributions, in a scenario with different characteristics due to the level of education.

1.3.2 Publications in chronological order

The results of our research have been published in different journals and have been presented in congresses as our work has developed. Figure §1.4 shows a complete list of publications in chronological order. Additionally we detail two works developed that are being considered for publication in journals and two videos showing the experiences when applying the experimentation phase with engineering students and, additionally, with the students of the school included in the research.

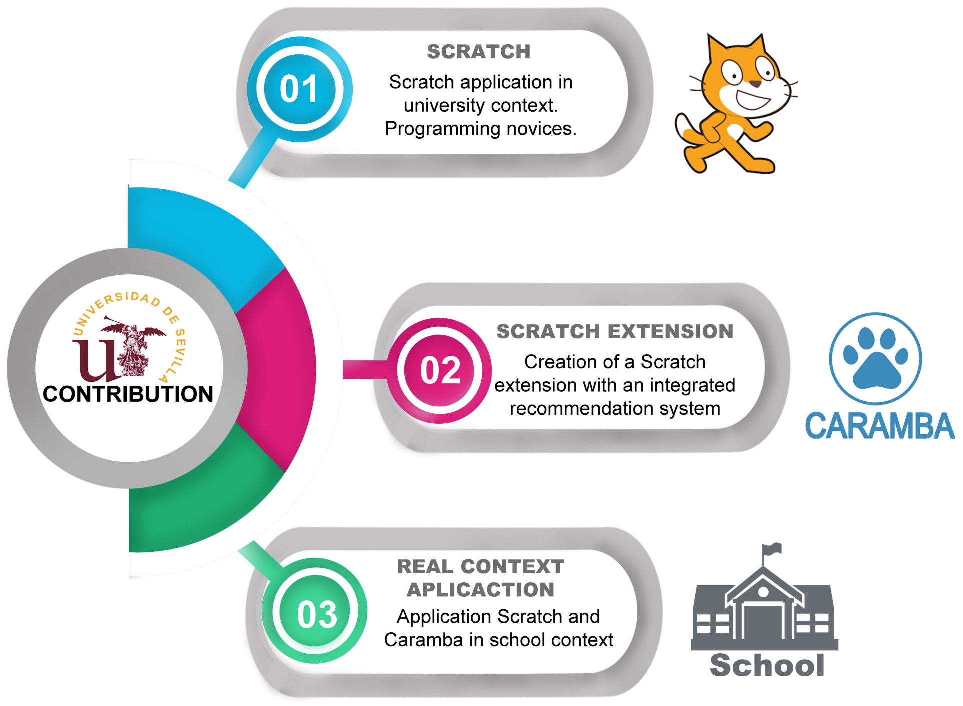

SCRATCH
01
Scratch application in university context. Programming novices.
UNIVERSIDAD DE SEVILLA
CONTRIBUTION
02
SCRATCH EXTENSION
Creation of a Scratch extension with an integrated recommendation system
CARAMBA
03
REAL CONTEXT APLICACTION
Application Scratch and Caramba in school context
School

Figure 1.3: Contribution scheme

Next, we present a summary of each published contribution in chronological order.

2016

- Conference paper in The World Engineering Education Forum & GEDC: this conference was held in Seoul, South Korea, where academics, deans of engineering faculties, students, and organizations, interested in contributing to the improvement of engineering education met. In that context we were able to expose an extract of the initial work of the research, we presented the first improvements in the academic results of the students at UNEMI, after the application of Scratch in the teaching of Pro-

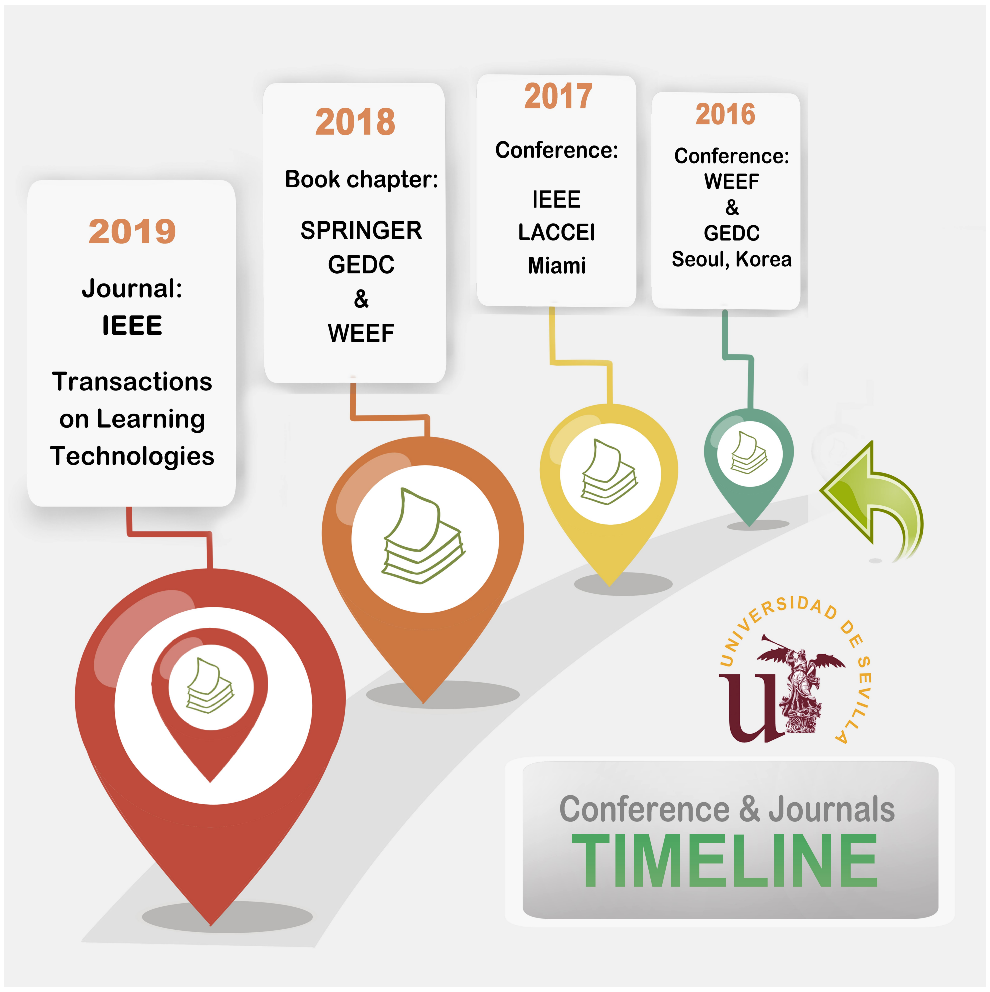


Figure 1.4: Timeline contributions

gramming for the first evaluation period of the subject FP.

WEFF & GEDC '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. Link: http://www.ifees.net/weef-gedc-2016-seoul/ Programming

2017

- Conference paper in Latin American and Caribbean Conference: Global Partnerships for Development and Engineering Education: Conference for Engineering and Technology. A paper on the work carried out with the students of the pre university at the State University of Milagro , Scratch + CARAMBA was presented, based on the best results obtained with the first semester of Systems and Industrial Engineering students. The intervention extends in the applicants for engineering This made possible to validate CARAMBA tool once again.the work carried out with pre-university students at Universidad Estatal de Milagro, Scratch + CARAMBA was presented. From the best results obtained with students of first semester of Systems and Industrial Engineering, the intervention of applicants in engineering programs is extended to students of engineering, who must pass the program course to acquire a place in engineering at UNEMI, this allows or validates once more CARAMBA tool.

LACCEI '17: J. Cárdenas-Cobo,David Benavides,Mayra D'Armas Regnault, Mariuxi Vinueza, Jorge Rodas. Programación con la herramienta SCRATCH +CARAMBA. Una experiencia de aprendizaje significativo. Proceedings of the 15th LACCEI International Multi-Conference. ISSN: 2414-6390. 2017. Link: laccei.org/index.php/publications/laccei-proceedings.

2018

- Book chapter Springer: The best papers of the WEFF conference & GEDC 2016, were selected to publish a book with the proceedings of the congress. For this purpose the article was extended as well as an epigraph which includes the improvement in motivation, with the help of a tool recommending exercises to be solved. Exercises are suggested based on comparison with similar users in terms of taste and difficulty.

  WEEF & GEDC '18: 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. Springer International Publishing, Cham, 2018. ISBN 978-3-319-60937-9. Link: link.springer.com/chapter/10.1007/978-3-319-60937-9_20.

2019

- Article in IEEE Transactions on Learning Technologies journal: Recommender Systems and Scratch is presented: An Integrated approach for Enhancing computer programming learning. In this work the book chapter is extended, adding the validation that allows to confirm the Hypotesis that the tool contributes to improve the academic results of engineering students.

IEEE '19: J. Cárdenas-Cobo, A. Puris, P. Novoa-Hernández, J. A. Galindo, and D. Benavides. Recommender systems and Scratch: An integrated approach for enhancing computer programming learn- ing. IEEE Transactions on Learning Technologies, 2.315 Impact Factor.

- Article in Computer and Education journal: The paper with the first phase of our experimentation has been submitted. In this paper, we explain in detail the application of an instrument to show the impact of applying Scratch on students' motivation. Jesennia Cardenas, Águeda Parra, José A. Galindo, Amilkar Puris, Pavel Novoa-Hernández, David Benavides. Using Scratch to improve learning programming in college students: a positive experience from Ecuador. Summit.

1.3.3 Audiovisual material

This section presents the audiovisual material developed for the diffusion of the work developed, as well as the user manual with all the profiles contained in CARAMBA.

- Engineering student experiences: This video was presented with the experiences told by the students after the experimentation phase. It was presented in the framework of the meeting of deans invited by Shenseng University, where proposals for the improvement of engineering education were shared. China Shenseng(2018). The video presented can be watched at YouTube: https://youtu.be/WqNMVUDG4ts

- Engineering students and the children experiences: This year a video with the experiences told by engineering students and the children at the public school where the tool was implemented after the experimentation phase was presented. The video presented can be watched at YouTube: https://youtu.be/qCAiM8O9hSw

- User guides: The user guides of the profile student, teacher and administrator can be seen in the video published at the following address Youtube: https://youtu.be/iPvZxMi4RSc

1.3.4 Degree thesis

After implementing CARAMBA in school environments(2019), in order to propose improvements to the tool, a degree thesis was proposed with a Computer Systems Engineering student from UNEMI, the project applied an evaluation based on ISO9126, the results can be reviewed in the repository: http://repositorio.unemi.edu.ec/handle/123456789/4748

1.3.5 Tool

As a result of this research CARAMBA platform integrating an exercise recommendation system with the Scratch tool was developed for autonomous learning . The platform allows managing different types of users (students and teachers) with their specific roles and obtaining statistics of student behavior among other things. This platform is available at Public IP: http://caramba.nas-code.com/ Further details of this application are given below in Chapter §5. The link in GitHub is: https://github.com/jcardenasc1/CARAMBA/tree/master/django_project

1.4 Structure of this dissertation

This document is organized as follows:

Part I. Preface: In the first part of this dissertation we present the main contributions of this research, as well as the background and motivational scenarios that drove this work.

Part II. Background: In the second part of the dissertation we explore and update the basic information necessary to understand the objectives of this thesis. We clarify the term not WEIRD and its relation to our work. In addition, the various pedagogical tools for learning programming are analyzed and the concepts of learning are discussed in depth, which contributes to presenting possible solutions in the following chapters.

Part III. Motivation: In this part, we review some ideas to improve programming learning and motivate our research work. We deepen into work related to the application of pedagogical tools for learning programming in university contexts and analyze the use of recommendation systems applied to education. This background also allows us to present the general and specific objectives set out in the contributions.

Part IV. Contributions: In this part we present the main contributions of this thesis. Divided into three chapters where it begins with the analysis of the motivation for the use of a pedagogical tool for learning programming in university courses, then the application of a recommendation system combined with a pedagogical tool and finally the application of the tools in other educational contexts.

Part V. Final Remarks: In this part, we present the conclusions of this thesis and propose future work to address new and challenging open research questions found during the realization of this dissertation.

Part VI. Appendixes: We will find the questionnaires used in the experimental phase, the tests applied in the evaluation of computational thinking, images of the extension to school contexts and statistical graphs referenced in the contributions.

Part II

Background

Chapter 2

Learning of programming

Nothing in this world should be feared...only understood. Now is the time to understand more, so that we can fear less.

Marie Curie

The teaching and learning of programming allows the development of computer thinking. From this premise, several studies of the pedagogical tools that allow a better learning to novice students have been carried out. However, there is little literature in developing country contexts.

In this chapter we present definitions related to the teaching-learning-programming process, the types of pedagogical tools for learning the program, we analyze how recommendation systems can contribute to autonomous learning and we define the non WEIRD context, necessary to justify our research work.

2.1 Introduction

-The programming process arouses a phycho-pedagogical interest because of its effects on the cognitive capacities of the human being. It also contributes to develop an intellectual activity that allows planning and designing strategies, analyzing problems, building algorithms, structuring instructions, understanding one's own programs or those written by others. In the same way, it helps to learn the syntax of the different programming languages, to compare strategies between expert and beginner programmers.

In our experience, many students have difficulties in the process of learning the programming disciplines, the most common obstacle has been the creation of algorithms according to the requirements of the class and the correct use of programming languages. This is because for many people, the high level of abstraction of the discipline and the difficulty to understand its complete conceptual structure constitute a serious impediment [63].

Algorithms must be expressed in natural language and people must have the appropriate tools to facilitate their development. In literature, several studies have been published in which proposals have been made to increase programming in education. Programming models or patterns, algorithm analysis, suitable environments for programming, among other research works have been carried out to improve the teaching and learning of this subject, such as the study developed by Gomes et al [31].

These tools are designed to help in the process of learning programming through graphic representations of the solution algorithms and/or the execution of a program [31]. The main research in this area has been carried out in developed communities (known in the field of psychology as WEIRD communities) [33], where the conditions for student learning are completely favourable.

However, there is little specialized literature on developing communities where students have to survive with very complex economic, social and technological barriers to learning. That is why in this chapter we detail pedagogical concepts, methodologies and learning tools that are considered the basis for our work and clarify the concept of non WEIRD communities, which we see as an opportunity for research.

2.2 WEIRD and non-WEIRD communities

According to Henry et al.[33] in order to better understand human psychologies, scientists need to stop carrying out the majority of their experimental research on Westerners (WEIRD), a term used to define Western populations, educated, industrialized, rich and democratic. Cultural psychologists have demonstrated that people from WEIRD communities tend to have a more individualistic and less communal outlook on life, whereas those from non-WEIRD backgrounds are more analytical and less holistic. [57].

Considering that university students from rich countries are numerous and also WEIRD, psychologists warn that this becomes a drawback, as they are not a representative group of human beings as a whole and are routinely used in studies to make broad, and most likely false, assertions, about what drives human behavior [40]. This is because it is easier to access WEIRD populations; studying people who live in non WEIRD countries can offer a substantially greater vision than just adjusting the instruments used in studied populations [69].

Studies analyzing ethnicity and post-secondary enrolment show that students from African American and Latin American backgrounds often start and/or complete their universities studies in a non-traditional way for example: late enrolment, part time university attendance and the rest from university attendance. Many students from ethnic minorities will first enrol on two-year courses in order to later graduate with a four-year degree [11, 28, 50].

Pope et al. [62] suggest that through the use of codified rules, humans can solve many problems with precision, and conclude that there are several aspects of cultural variation, knowledge, environmental uncertainty and educational training, which could emphasize the intercultural differences observed in the use of flexible strategies, so the alternatives allow them to be more efficient. The authors reported that, the results in Western communities are not the same as in non WEIRD communities. This is because people, after having chosen a suitable solution will ignore any other options, even if those other options are more effective.

Arnett et al.[4] in their work state research papers published in the top psychology journals focus on Western societies that are not representative in humankind as a whole, this coincides with the work of Henrich et. al.[33], which presents the results of an analysis of ten years production in psychology journals that suggest a similar bias in the field of research.

Usually the first authors of the publications in the major journals of the

Characteristics	WEIRD	non WEIRD	References
More individualist social orientation	√		[57]
Less collectivist social orientation	√		[57]
More analytic cognition (equity, cooperation, punishment)	√		[33, 57]
Holistic cognition (moral reasoning and independent/interdependent self-concept)		√	[33, 57]
Representative study group		√	[40]
Easy to access populations	√		[69]
Nontraditional university enrolment patterns		√	[11, 28, 50]
Precise solution to problems	√		[62]
More publications in journals of Psychology	√		[4]
Reasoning strategies	√		[33]
Implications for such traits as motivation or emotions	√		[33]

Table 2.1: Differences between WEIRD and non-WEIRD populations

American Psychological Association (APA) are affiliated with academic institutions of WEIRD societies, representing 12% of the world population. These publications are accessed mostly by WEIRD populations and the authors as a norm extend their results easily to all humanity [4] nonetheless, Henrich et al.[33] demonstrated that there 's a great difference between WEIRD and non-WEIRD people from the cognitive point of view and in the process of making social decisions(for example, equity, cooperation, punishment), strategies of reasoning (in which WEIRDS have a tendency to be more analytical and non-WEIRDs to be more holistic), moral reasoning, independence and interdependence (whereby WERDS are often more individualistic, effecting motivation and emotions.

The dominance of WEIRD countries in psychology is due to economic reasons, Arnett [4]. In developing countries, governments usually allocate funding to more critical expenditure of to psychology research. Nonetheless, this fails to explain the low numbers of research papers from developed countries that are not members of WEIRD (for example, Japan).

Studies in the field of psychology warn the need to carry out research to obtain replicable results for non-WEIRD communities, which have characteristics different from the WEIRD populations (see Table §2.1). This would allow to reproduce experiments in different contexts, through validated measuring instruments [62, 68] to ensure results.

2.3 Programming teaching learning

The inconveniences experienced by students during the initial stages of their programming studies have in many cases led to frustration and even failure at school. In fact, introductory programming courses have a high dropout rate and students usually do not learn to programmer well [51].

Several factors affect the students' performance in introductory programming courses. However, the key factor is that students have a weakness in cognitive reasoning skills and in applying problem solving strategies, i.e. they have not developed the cognitive abilities to the level they need to start learning programming [51]. In literature, there is an extensive research at the international level to examine experiences in the teaching-learning process of programming in search of the most common difficulties students face during introductory courses at university levels, and how to address these difficulties. Some relevant studies are mentioned below.

2.3.1 Pedagogical initiatives

A study developed by Lahtinen et al. [49] with university students and professors to identify difficulties in programming learning in introductory courses, where more than half of the students (58.6 %) who participated in the study already had programming experience when they began studying at university and only 40.6 % considered that they had more or less moderate programming skills, they found that:

- The wide range of experience levels among programming students becomes a problem when it comes to designing, interesting and challenging teaching for all.
- The most difficult factors in programming were to understanding how to design a program capable of solving a given problem, dividing functionality into a set of procedures, and locating errors in their own programs.
- Self-study exercises were considered more useful than lectures and hands-on sessions in computer classrooms.
- Self-programming was rated as more useful than self-study.
- The biggest problem among new programmers is not relate to their ability to understand the basics, but to their ability to apply their knowledge.

From these findings we can highlight that student learning will be greatly improved by learning through practice and real life situations. While theory is essential to programming, students also need to put theory into practice to fully understand the concepts.

For example, Wing et al. [80] refer to computational thinking as a kind of mental activity in which people state a problem to be solved with a computer. Either humans or machines, generally a combination of both, can carry out the solution.

It is worth noticing the experience of De Kereki et al.[27], who obtained improvements in the results of initial programming courses and a decrease in student dropout by incorporating a set of activities to promote active learning classroom and by emphasizing teacher reflection on the importance of including variety and innovation in teaching, with the aim of encouraging active learning and keeping the students motivated for the content.

Questions have been raised about other aspects to consider in the development of introductory courses. For Moroni et. al. [58] if we starts from the need to follow a structured methodology based on the imperative paradigm, we must consider: how to accurately express the problems to be solved, the solution strategies to choose from; how to translate the solution into an algorithm; and, how to write the corresponding program into a programming language. Moroni et. al., argue pupils the need to develop effective problem solving abilities. Also, the methodology used in this learning process should tend to solve increasingly complex problems.

Gomes et al. [31], proposed a computational system that helps to reduce current learning difficulties, through an environment based on prioritizing students' learning styles in the design of different problems and tasks. On the basis of a constructive approach the system provides opportunities that should be developed while the learning process is taking place. They make trials, deduce, and step by step structure their own knowledge. This creates a possibility for a more effective, personalized interaction with them, both in the definition of meaningful activities proposed to the student, as well as in the information given by the context.

Even though programmers need to be creative, and programming needs creativity, the number of teaching methods which are getting closer to the creative and potential role of programming is low in computer science research. Nonetheless teachers should foster the students' motivation to improve their creative skills. Researchers and computer education teachers, as a rule do not emphasize creativeness, they focus more on the syntax and in making the program meet its goal.In that context for beginners programmers should interpret and give solutions to problems algorithmically which are very difficult to solve. That is also the case of encrypting the code of an algorithm in a programming language which constitutes a challenge as well.

The learner must develop a self-learning capacity necessary to perform according to the requirements of today's society. To this end, For Hernan et al. [34], propose to change the teaching focused on the transmission of formal contents (expository classes), to a teaching focused on competence development and based on the resolution of problems as close as possible to reality. Problem based learning improves retention in the long term by forcing the learner not only to understand but to apply or reflect the learned concept.

Human cognitive system has been a subject to many studies. Caspersen et al. [14] presented a model for human cognitive architecture and three learning theories based on this model: cognitive load theory, cognitive learning and practical examples. In this study they designed an introductory course to object oriented programming, the main techniques applied were: worked examples, scaffolding, discolored orientation, cognitive learning and emphasis on patterns to help create schemes and improve learning, it also uses a characterization of programming based on models. Since then, several authors cite this work [29, 35, 73, 77].

2.3.2 Bloom's taxonomy

Bloom's taxonomy describes and categorizes particular topics by how cognitively difficult they are to learn by placing the topics in a hierarchy from less to more complex. It states that a student must master one level before passing to the next. [36]. The above statement was verified by Kranch et al. [46] in their study where they compared three instructional sequences for teaching programming concepts to novices. According to the results, varying the sequence did not change what novices learned, but it did affect the difficulty they experienced in learning it.

There are six categories in the taxonomy: knowledge, comprehension, application, analysis, synthesis and evaluation, whereby knowledge is the first level and is relates to memorizing information and definitions. As one goes up through the levels, one must make greater use of knowledge and mental capacities until one reached evaluation, the highest level of cognition, which is relates to creating, developing and writing ideas[73].

Shuidan et. al.[73] showed in their study that teaching from basic to abstract produced the lowest rated effort and difficulty and the highest efficiency.

Beginner programmers need plenty of opportunities to practice using the fundamentals of programing syntax and structure before they are ready to tackle standing program solution plans. This should encourage confidence among less-advanced beginners in introductory courses.

Learning to program is difficult and it is usually responsible for high dropout rates in computer schools, because newbies aim to understand programming early on and when they do not, this leads to frustration and finally leads some to give up [73]. In this study they report on the efforts of the novice programmers in the summative evaluation. We analyzed the answers provided by the beginners to the questions of the final exam, and we tried to understand why students make such mistakes, through applying Bloom's taxonomy to determine the skills obtained, in conceptualization, analysis and application.

In a study by Walley et.al. [78] which aimed at investigating how beginners understand the exercises planned to develop a program, a set of questions was designed to test the full range of cognitive processes within the program. Understanding the cognitive processes of Bloom's Revised Taxonomy, they found to be more difficult than expected although the revised taxonomy allowed them several ideas to generate the questions they found difficult to locate at each level of taxonomy, because the examples were not so simple to translate to programming and it was also difficult for them to match the cognitive tasks performed for each question with Bloom's cognitive processes..

Despite the evaluation of their application in different languages, the differences in the initialization values and in the relational operators used by the students, did not result in changes in the logic of the questions or answers at the end of the study.

The experiences reflected in literature imply that traditional approaches and methods for teaching programming may not be very effective for a lot of students. Rather, teaching must be adapted to the realities and cognitive needs of students for their growth as programming professionals.

2.3.3 Patterns-based learning

Programming models or patterns are simple design patterns that can aid beginners, because they are solutions to simple and recurring algorithmic problems. The models are like building blocks that allow one to construct new programs. These solutions appear often in solving calculation problems and are similar to programming strategies that have been designed by professionals. Therefore, students take advantage of good programming practices through the application of programming patterns. Various computing fields have adopted design patterns, including software engineering, teaching computing, man-machine interaction and e-learning..

Despite the general search for ways to develop programming competences, there is no consensus on what is the best way to learn how to program. Aquino et al.[2] propose an alternative improvement in the learning of programming models (or patterns) using video-games, as they point out that there is no pedagogy to teach programming that stands out significantly from others. In addition, they indicate that there continues to exist the need to create new pedagogical methods for introductory programming. In their article, these authors present a pedagogical approach in support of creativity in the programming and the results of a successful case study, in which the instructor enables the students to learn programming patterns by means of a game. This game used specific materials and improved students' motivation and learning.

Referring to the benefits of this alternative, Aquino et al. [2],point out that by means of drawing programming (programming patterns), students can combine and recombine a repertoire of building blocks in a limitless way to foster creativity in themselves. In addition, they get to know new possible solutions, which enables them to focus more attention on the process of solving problems and in our opinion leaves aside the fact that the novices are more concerned about their problems in syntax when using programming languages directly.

2.4 Tools for learning programming

The current context of teaching-learning processes is characterized by the progressive inclusion of Information and Communication Technologies (ICTs). This section provides a description of a number of ICT tools that are being used successfully to support the teaching and learning of programming, and have yielded positive results, not only in quantitative but also qualitative terms. Each educational tool has proven effective in particular aspects, so the use of ICTs should be to find the appropriate technique, method or model to meet the specific needs identified [20], and to support the teacher's performance, so that students learn the proposed objectives.

The proposal of new developments for improving programming teaching has increased in recent years and several studies refer to the tools that have been tested for use in programming teaching at introductory levels. Gerrero et al. [32] and Silva et al. [19]. Other people involved in the programming process have reviewed the literature based on educational tools for POO. Georgantaki et al. [30]. Nowadays, there are many visual programming tools available for teaching this subject. Each tool has been designed according to the particular characteristics and needs of the target user at a given time. The different visual tools can be analyzed according to relevant parameters or characteristics that must be considered to make the decision about their application.

Teaching systems were designed in order to facilitate the learning of programming. These systems have tried to render programming accessible in three main ways. Firstly, they simplify programming mechanics; secondly, they offer support to students; and thirdly, they motivate learners Kelleher et al. [43].Programming software developed for children and adults, who have no programming experience, usually includes tools that allow them to delve into basic programming concepts.

Systems have focussed on rendering programming mechanics more manageable. For example, they have taken out unncessary syntex, introduced programming in visible contexts in which learners can view immediate results and explored different options to typing programs. The authors mentioned above, presented a proposal to classify the different programming teaching systems available, by the primary aspect of programming that the system attempts to simplify. Their classification covered forty-eight systems.

To counteract such obstacles during learning, the tool to be used for teaching should avoid syntax errors; allow the student to develop logical and creative thinking, by performing various activities (for example, games); and allowing the visualization of the execution of algorithms. Blocks-based Programming Languages (BBPL) comply with those characteristics. Block programming environments have become a popular way of introducing coding and as a building block towards traditional languages based on text, however one can also use these environments to write ”real code” [6].

Willen et al. [79], consider that the following properties can be used to approach the definition of BBPL:

- The interface used by the language contains building blocks (virtual and/or physical).
- It allows to develop programs through the combination of blocks of atomic or compound code.
- Element code blocks can contain other blocks.

The interface that requires the first property must contain blocks of selection code, representing building blocks that can be selected through input devices.

Understanding programming requires the application of computational thinking.Much research has been done in this area. Segredo et al. [72] conducted a comprehensive research with the aim of determining already-available initiatives, projects and tools that aid students to think in a computational manner. The following feature dimensions were selected for the comparison: Available for free, Online tool, Online repository availability, Project reusability/remixing, Learning difficulty, Block-based/Text-based/Both, and Target programming language.

Regarding the introduction of new programming environments in educational contexts, Anfurrutia et al. [1] recommend considering that, first, no single programming environment is befitting to all situations, and second, the activity carried out through the tool has to be appealing and relevant to learners. In order to increase the programming education's effectiveness, it is essential to select the most appropriate programming language, taking into consideration the learner's cognitive level and the reason why they are learning to program[72].

Concerning the purpose of learning programming, programming can be learned by two main reasons: to become a professional programmer or to achieve other goals through creating programs [6]. Young people who start their university studies in computer careers enter the first group mentioned. Regarding their cognitive level, those who have not acquired programming knowledge or experience during the previous school stages can be considered at the initial learning levels.

Many proposals are built in answering to people's needs and considering their level and purpose for learning programming. Yoon et al. [83] derived the educational programming language (EPL) selection criteria for beginner computer science by means of the Delphi method using a panel of 20 experts. The objective of their study was to increase teachers' ability to choose a suitable EPL for learners, taking into account the learners' characteristics and transitioning programming knowledge from the EPL to a general-purpose programming language.

Teachers cannot easily adapt tools providing to their own needs, those who provide what to do in the next step, they rather do so without recognizing alternative strategies. Keuning et. al. [44] perform a systematic review of the tools to learn how to program, the training feedback they have to help students improve their work. They also reviewed techniques used to gain feedback, analyzed how adaptable feedback is and how people can evaluate these tools. They report on the encoding results of a total of 101 tools and reveal that the comments focus mainly on the identification of errors and less on the solution of problems and on the next step to make.

Software visualization is used to aid the programmer/user of a program to understand the artefact being observed. It is a relatively young development with many advantages being made in the 1900s. Software visualization environments are developed as programming tools for professionals or instructional tools for demonstrations or interactive study in Costelloe et al. [16].

2.4.1 Visual tools

The first tools for learning programming were developed back in the 1960s. Concretely, in 1967, Logo was developed using the Lisp programming language [61], which allowed for the implementation of I/O operations, iteration, data structures and algorithms. Nowadays, the programming challenges and requirements for graduated students have increased interest in improving the tools and techniques to teach programming, such as Scratch [66], Greenfoot [39, 48, 75] and Alice [15, 82]. They can be classifed as text-based and block-based [22], the Table §**??** shows the results of systematic review of programming tools, in this paper [74] the challenges are examined that new programmers must deal with and popular online methods for helping students.

Currently, special visual environments for Visual Programming are being used. And there is a consistent research on the matter. According to Kaucic et al. [42] early research results indicate that visual programming is more effective than traditional textual programming, because it is more motivating, less boring and does not require the syntax of programming languages. Willem et al.[79] mention the different terms used in literature to identify them. Terms like "Graphical Programming Languages", "Visual Programming Languages", and "Block-Based Programming Languages" are being used to represent similar programming languages.

Guerrero et al. [32] establish a typology allowing to properly organize and study the large number of tools used to assist in the teaching-learning process of programming. The authors call "Visual Learning Tools" those tools aimed at graphically representing a program, since the algorithm of the program and its development can be observed through graphical components. They are tools that allow a high degree of interaction with the student. In addition, Integrated Development Environments (IDEs) for their characteristics such as syntax coloring, compilation, debugging, contextual support, among others, help the learning process depending on the techniques used.

The use of pictures and animations as educations aids is an accepted practice. Visualization is used to clarify complex concepts and to enable students to develop mental models of the underlying concept, an algorithm or the steps of the execution of a program [16]. Kunkle et al. [47] in their study make reference to previous research on POO learning through BlueJ. Findings indicated that although BlueJ helps learners to comprehend object-orientated programming concepts, it fails to help them comprehend the 'flow' of a program. The authors' findings imply that the learners could not trace the program's source code and therefore could not understand the program's execution.

Alice, developed by Carnegie Mellon University, works in an animation environment of a 3D virtual world, that through the creation of simple animation or video games teaches students the basic constructions of programming. The most interesting thing is that each instruction in Alice is equivalent to the declaration of programming languages oriented to objects like Java and C++. So Alice contributes to learning the fundamentals of cite object oriented programming [76].

The goal of the Alice is to enhance students' initial experience with learning to program in two ways: remove needless frustration, and provide an environment in which novice programmers of both genders can create compelling programs. Needless frustration is minimized by allowing students to create programs without ever typing a line of code, thereby preventing syntax errors. Programs are built primarily by dragging and dropping objects, and include associated methods and standard control structures with the editor. [47].

Scratch is a 2D environment that supports learning through the creation of animations, games, and interactive stories, developed by the MIT Media Laboratory (2003-2007) in agreement with the University of California Los Angeles. Its environment presents drag and drop functionalities, and allows using objects called sprites and blocks that define a logical operation to perform. It allows to improve the learning of such concepts as: variables, operators relationships and logic, and data types.

Scratch is a popular option in this review, considering that the trend is moving toward an intelligent tutoring system, where students have a personalized commitment to their learning experience. The tools are presented as an option for problem solving in an integrated environment and block programming, they also focus on: supporting instructors on the curriculum, challenges in CS and active learning through online tools.

Greenfoot is also considered within visual tools as a 2D virtual world. It is a highly Specialized Educational Environment for Developing Interactive Graphic Applications. It is based on Java Programming language [9]. It allows the learning of fundamental concepts such as inheritance, abstraction, polymorphism to a higher level than introduced in such environments as Scratch or Alice, and requires the introduction of Java code to a basic level.

BlueJ is an integrated development environment aimed at teachers as learners introductory object-oriented programming in Java [47].It consists of a graphical user interface for Java that allows the visualization of the programming for beginners. It includes a frame based editor called Stride that allows block programming such as Scratch and Greenfoot. You can experiment with objects, classes, and create methods before moving to conceptualization and without the need to know Java in depth. In the improved version you can share projects through Github(social application on the web)[8].

Turingal consisted of a Turing machine and a magnetic tape. The student wrote instructions using a language based on Pascal's syntax, which allowed the Turing machine to write and read symbols through the magnetic tape. For Bubica et al. [9],Karel J Robot and Turingal systems sought to make programming more specific or to reduce programming abstraction.

There are tools derived from previous generations that improve their interactivity, among them PLM (Programmer's Learning Machine) which allows to visualize the execution of code step by step for different programming languages [32].

A lot of research is done to help users to build their own blocks. Bubica et al. [9], for example, mention Snap, previously known as BYOB, which is a visual drag-and-drop programming language and an expanded re-implimentation of Scratch. Snap includes first class lists, first class procedures and continuations, making it appropriate for introducing high school or university students to computer science.

Some programming languages have not lasted in time. The first programming languages were very difficult to use and many could not learn the syntax of programming [66]. This is why programming tools such as Scratch, Alice, Greenfoot, have tried to overcome these limitations, since they are visual languages and there is no need to write programming lines, therefore typing mistakes are avoided; all types of customized projects and activities can be carried out using multimedia resources [52].

Alice, Greenfoot and Scratch environments particularly stand out, as environments that give students help they need, even though they were not developed at the same time or within the same context, these environments can be classified together because they share common characteristics. All three are visual environments are intended to encourage direct involvement in any attractive activity and all three have the task to introduce students to programming [9].

2.5 Scratch

Scratch is an environmental programming which was built by Lego's constructive learning ideas of Logo and was developed by a group of researchers from the Lifelong Kindergarten Group of the Massachusetts Institute Media Laboratory of Technology (MIT), under the direction of Mitchel [65]. It is a free software, which runs under various operating systems and can be easily installed, which was created in order to develop the logical and mathematical thinking in children and young people.

Scratch is composed of a dynamic, attractive, colorful and simple graphical interface that allows to perform, animations, games, dialogues, simulations, diverse activities and interactive comics or other programs that often arise from the student's own creativity and can be shared with other students or users of Scratch. Its environment uses menus, controls called color differentiated block palette that allow the design of the program created by the student, these controls come together as a puzzle in an orderly and logical way for the program to work properly [38]. One of the benefits of Scratch is that it is visually attractive and fosters learning in an active manner [45].

To be more effective in educational environments, programming tools need to be very interactive and socialization oriented. Scratch is a social computing environment and also a programming language that boasts a well-supported interface, according to Malonely et al. [54] Although initially targeted at teenagers, i.e. a younger audience than that in the first year of college, interest in Scratch is increasing at university level as a way of introducing learnings to difficult programming concepts.

Scratch has programming languages and its integrating capability is very important. Programming languages must have the ability to encompass different types of projects so that people with very diverse interests and styles are encouraged to schedule.Resnik et al. [66] The creators of Scratch have taken into account three basic principles in the design of this programming language to enable them to achieve this capability. These principles are: that the programming language be playful, meaningful and social.[66]

- Leisure programming language. A programming language facilitating gaming and feasible to try out different options easily. For the authors the way to program in Scratch is similar to the way to build with Lego. Lego construction pieces have connectors that suggest how they go together with each other and make easy to fiddle with them and start building objects. Similarly, Scratch has some "programming blocks" of different colors, with connectors allowing them to fit together. The goal is for children to play with them from the beginning and try to build simple programs [66].

- Significant experience. One of the principles of learning is that people working on meaningful projects learn and enjoy. The creators of Scratch have given priority to two design criteria: diversity (which can support different types of projects: stories, games, animations, simulations) and customization (that projects can be customized by importing photos, voices, graphics, etc.)

- To promote social interaction. Scratch has a large community of people who is attached to its website, and who collaborate and build on the work of others. Other people will support, criticize and build on others people's projects, the goal is to finally turn out an interactive and enriching learning experience for all [66].

Given the difficulty that novice programmers have in understanding programming concepts as well as the same process of learning to program, Koorsse et al [45] studied the impact of programming support tools (PAT for its acronym in English: Programming Assistance tools) that beginner programmers can utilize in order to learn how to program and/or better comprehend programming in secondary schools in South Africa.

This research's main goal was to establish to establish programming support tools using a PAT affected students' ability to understand programming concepts and motivated them in terms of programming in general. In the results of this research, no conclusive evidence was found that students using a PAT had a better understanding of programming concepts and were more motivated towards programming than students who did not use a PAT, In other words, the quantitative analysis suggested that the PATs did not affect students' ability to understand programming concepts in general.

However, in a case study of Scratch by the National Center for Information Technology and Women [5], Scratch is described as a tool for "promising practices" to increase gender diversity itself. The research concludes that the tool uses hands-on, active learning, its graphical interface is attractive; it motivates creativity; it allows feedback; and it avoids syntax errors, so that users can concentrate on relevant things such as processes and concepts.

They conducted two case studies. The first case study was a parallel network of ordination. In unplugged activity, children move through the network, and when two reach a node they compare the values of their data, with the smaller one taking the path to the left and the other to the right. After passing through the network, they leave the established order. This activity requires key comparisons, which are common in many fundamental algorithms, but also makes use of the animation that is natural in Scratch and Alice, to emulate the Unplugged version where children physically walk through the network

The second concept dealt with general ordering algorithms, choosing the type of ordering. This concept requires languages to be able to easily use matrices and their basic functionality, and to display operations in the matrix. The final concept was to convert binary numbers to decimals, which requires working with bits instead of integers. Authors conclude that Scratch and Alice represent a creative and attractive way to teach programming and can be used for important computer concepts. Scratch provides an easy way to use environment, but lacks features such as sprite arrays, simple parameterization and a larger virtual world size, can restrict what can be implemented, and how it can be implemented.

A study by Malan et al. [53] showed that there was a decrease in the number of students dropping out of computer introductory courses when Scratch was used to introduce programming concepts to students. One disadvantage of Scratch is that users might have difficulty moving on to a traditional programming environment, unless they also use a middleware tool that links the concepts and programming methods that Scratch has introduced to them with a programming language where syntax becomes relevant [66].

Scrath focuses on syntax rules due to the use of the drag and drop interface. Therefore, users don't need to worry about overloading the syntax of the programming language, but focus more on planning the programming solutions [45].

The experience of the leading universities in the world in teaching programming is a point of reference for many researchers. In Harvard, students are trained in Scratch for a week, and during that time they internalize complex concepts, such as event management, concurrence, threads, as well as the most trivial fundamental concepts such as types of repetition, selection, variables, logic's and data types. The conceptual fundamentals that are immediately accessible to students through Scratch make the transition after a single week in C or Java more natural.Wolz et al. [81]

The transitions from one environment or condition to another, in the study of programming aspects are also a focus of interest. Armoni et al.[3] looked at how students were able to transition from studying CS in Scratch's visual environment in middle school to studying it using a professional textual programming language (C# or Java), whilst at secondary school. The case study discovered that students who had used Scratch were better able to learn more advanced concepts in secondary school and could learn them faster. Although by the end of the process there were no significant differences in achievements between Scratch users and non-Scratch users, the former were more likely to enrol in CS classes and had higher levels of motivation and self-belief. Therefore, there was a benefit to teaching CS, particularly visual programming, in middle schools.

2.6 Autonomous learning

According to McCombs et al. [55] learners' inclination for self-study in terms of processing, planning and monitoring learning activities as well as motivating themselves. The latter is essential for the former to take place and motivation, in turn, depends on students believing that their efforts will bring about academic success. Students need to be capable of evaluating their ability in a positive way.

Two different conceptual approaches are used to refer to the student's own learning process, sometimes referred to as synonyms in literature: self directed and self regulated learning. Rauner et al. [64] Many studies aim at clarifying these concepts Ferrer-mico et al. [25] analyzed how programming in Scratch affected young apprentices' ability to self-study and concluded that most students were aware that their knowledge of the subject could improve if they spent more time utilizing Scratch.

In a study conducted by Chia et al. [24] the influence of setting prior goals and regulating learning was investigated, the results revealed the following: nonspecific objectives were more efficient that specific ones in terms of promoting resource management related to self-regulation, so a self guided learning with goals set according to user profile.

In university education, distance education, vocational training or computer technology are the areas in which autonomous learning is being included with the highest priority, however to develop this model of teaching, it must be recognized that autonomous learning influences the learner to assume responsibility and internal control of the personal learning process, it is also known as self directed learning, that is, a type of learning where the standard is set by the subject learning [23].

University education should be oriented towards a comprehensive development of the person "a more humanistic cooperative, researcher and reflexive type of person". For this it is necessary to train students in areas of cooperative and autonomous nature, for which teaching models which would promote a new style of learning will be needed. Escribano et al.[23]

Responsibility and self-motivation for learning, within the context of Ecuador, where students come from rural sectors, where they mostly lack technological tools to support teaching models, with low internet access in high school educational establishments and students accustomed to classrooms, directed by teachers with behavioral methodologies, The motivation for research and development of autonomous learning is limited.

An example of this is the one presented by [7] in a study of university students taking a programming course, which coincides with the previous paragraph in considering that the moments when the teacher has control within the classroom are those that mostly meet the learning requirements, besides being the ones of greatest pleasure for the students, they like the practice, but not the theory. Autonomous activities are the ones that students like the least, this shows that there is lack of motivation, for self-learning. It suggests new methods for maintaining and increasing motivation for work and tasks outside the classroom.

It is inevitable in the current context, where access to information has increased, that students acquire a new role, different from the customary, they must worry about learning to learn, adapting themselves to change, transforming improved reality by managing their knowledge and becoming a meaningful and autonomous learning agents, all this is considered basically necessary for the acquisition of an academic and intellectual quality training [12].

But what does it mean to learn to learn?, for Diaz et al. [21] is to teach students to become independent and self regulated apprentices. However, at present it appears that what the curricula at all levels of education promote are apprentices who are highly dependent on the educational situation, with ample or limited conceptual knowledge of different disciplinary subjects, but with few tools or cognitive instruments helping them to face new learning situations belonging to different domains and useful to the most diverse situations.

One of the types of products to be developed in an autonomous work among others is the creative resolution of problems or cases, which we can be integrated through the methodological approach of project oriented learning, pedagogically based on the constructive approach of Kolb's experience cycle (1984). It is oriented to autonomous learning in which the research process and the elaboration of a work, focused on the creation of a product or the appropriate resolution to a problematic situation, acquires special relevance, through a range of tasks, the application of interdisciplinary knowledge, and the effective use of resources. It allows solving problems with the search for open solutions thus giving the student the opportunity to generate new knowledge (Itesm, 2004) [26].

2.7 Summary

Based on studies revised, the criteria considered and the needs of university students at introductory levels of programming, it is determined that the Scratch tool is the best alternative for the first meeting of young people with programming, who due to their lack of previous experiences in the matter, and even in computing, need a low learning difficulty tool. In addition, the tool is known for giving great emphasis to the development of computational thinking so necessary for students to start their training as professionals in computer sciences.

The change of learning methodology through a strategy of knowledge self-regulation, suggests that the student should be attracted and motivated to the use of a certain learning platform. Scratch has proven to be a tool that puts syntax in the background, instead prioritizing the result and the ease of assembling pieces of code allowing showing results of a given action.

Part III
Contributions

Chapter 3

Motivation

There is a driving force more powerful than steam, electricity and atomic energy: the will.

Albert Einstein,

Learning programming in university contexts, especially for young novices with little access to technology, is a recurring problem in Ecuador. To improve this difficulty, a number of pedagogical tools for teaching programming have proliferated. For example, Scratch focuses its use on concepts and not on syntax, and despite being a tool created for children, its application with satisfactory results in universities has motivated its analysis in our work. There are few cases found in universities that generate a research opportunity.

In this chapter we present an analysis of existing VPLs for the development of programming skills, we have reviewed cases in university environments, and how recommendation systems can be applied in educational environments to motivate autonomous learning. [†1]

[†1]Part of this chapter is published on IEEE Transactions on Learning Technologies [18] and Springer International Publishing [13]

Chapter 4

Scratch as a supporting tool for learning programming

"Science serves to give us an idea of how enough our ignorance is."

Robert de Lamennais

Developing programming skills is a complex task for university students from non-WEIRD contexts. This chapter presents the empirical results of the experiments we conducted to determine the impact of a VPL on programming learning. We have tested a number of hypotheses related to the use of Scratch as an autonomous VPL.

Chapter 5

Improving autonomous learning with Scratch

"I have deeply regretted that I have not advanced enough to understand the great fundamental principles of mathematics, because the men who dominate them seem to have a sixth sense".

Charles Darwin

As a result of the first experimental phase, it was necessary to establish a VPL that would be more attractive for university students to learn programming. This chapter presents the empirical results of the experiments we carried out to determine the incidence of the VPL Scratch, extended with a system of exercise recommendations, called CARAMBA(scrCrAtch RecommendAtions for Mastering BAsic Computer Programming).

†1

†1 Part of this chapter is published on IEEE Transactions on Learning Technologies [18] and Springer International Publishing [13]

Chapter 6

Applying CARAMBA to transfer results

Learning to think in a new way

Manifiesto Russel-Einstein the 1955

Students who access higher education come from educational environments with little or no access to technology, turning programming into an entry barrier to university. This chapter presents the results of extending Scracth + CARAMBA to another school sector to develop the computational thinking of its children.

This chapter presents the results of the extension of Scracth + CARAMBA to another educational sector to develop children's computer thinking.

Part IV

Final remarks

Chapter 7
Conclusions and future work

Defend your right to think, because even thinking wrong is better than not thinking.

Hipatia

This dissertation presents the difficulty of novices from non-WEIRD contexts in learning programming and how the impact of a VPL can improve the results. This chapter presents the conclusions and future work based on the results obtained.

7.1 Conclusions

In this dissertation we have shown that:

To improve the academic results of the novice students of programming of the Faculty of Engineering of a University where their students belong to non-WEIRD communities through the innovation of formal education to a self-guided teaching and assisted by a tool created to recommend their route of learning is not only feasible to achieve, but it has been able to evolve to adapt to other contexts where, despite maintaining the non-WEIRD feature, the variable is the age of the apprentices.

Although the proposal was focused on contributing to university learning, it now successfully supports the learning of children between 8 and 12 years old. In addition, as research problems deviate, new research challenges arise. This motivates the need for new research and new solutions. In this thesis we are confronted with some problems derived from the new tool of automated analysis of recommendation for learning. Specifically, we have validated the change in motivation to learn autonomously in this thesis. To demonstrate the validity of our work we have validated in different contexts and stages, from the first semesters of education, continuing with engineering aspirants and extending the work to a public school with 460 boys and girls between 8 and 12 years.

- Q1: Did the students really use CARAMBA?
- Q2: Did they practice with exercises recommended by CARAMBA?
- Q3: To what extent did the level of practice in a given learning concept influence student success?
- Q4: To what extent did practicing with CARAMBA contribute to student success?

The results obtained allow us to conclude that: The main cause of the success of CARAMBA is the high level of practice of the students with this tool. However, one question still unanswered is: what causes students to exhibit this high level of practice with CARAMBA?

In our opinion, this is mainly because the students are very motivated with CARAMBA. We have observed that today's students, like the majority of modern society, are closely linked to ICTs.Therefore, by using CARAMBA $\rightarrow$ more motivation $\rightarrow$ more practice $\rightarrow$ better chances of success.

Research opportunities

We recognize that the hypothesis: Use of CARAMBA $\rightarrow$ More motivation $\rightarrow$ More practice $\rightarrow$ Better chances of success a requires an in-depth formal study to explain it. Our future work will be oriented in that direction.

7.1.1 Discussion and open challenges

In the Chapter §1 we detected and enumerated the research questions we were willing to address in this thesis document. Table §7.1 shows the chapters where we target each question with the contributions we published. Also, in the next paragraph we will go through all different research questions and explain how we addressed them.

RQ1 (Motivation): To what extent will Scratch's learning strategy affect the students' motivation in FP? We presented and evaluated motivation in Chapter §4.

RQ2 (Concept Learning): To what extent will Scratch's learning strategy and an exercise recommendation system affect the students' ability? In chapter §5, we studied the impact of an exercise recommendation system on the students' autonomous learning .

RQ3 (Final Performance): To what extent will Scratch's learning strategy and an exercise recommendation system affect the final performance of students in FP? The impact of the application of the tool in other contexts, in addition to the university and its impact on the perception of social responsibility that it generates in university students, in addition to the motivation to learn to teach, is presented in Chapter §6

Research question	Chapter	Published contributions
R1	§4	–
R2	§5	[13, 17, 18]
R3	§6	–

Table 7.1: Links between chapters, research questions and publications

Then, back to the original question:

How to reduce the failure rate in the subject fundamentals of programming?

When analyzing the previous research questions, we can affirm that it is possible to apply a recommendation system to contribute to programming learning, we have evidenced an improvement in academic results in terms of grades and the result of the interest aroused by the tool on learning, as well as favorable results in the evaluation of computational thinking through an instrument validated in other contexts and that has adapted to that of our field of study.

To answer the original research question, we summarized that we improved the failure rate of students in the Fundamentals of Programming course. Specifically, we focused on awakening motivation for learning through a tool and then evolved to autonomous learning methodology guided by a recommendation system that allows interaction with similar learning peers... However, there are still some facets of research that need future work.

7.2 Future work

In addition to the future work mentioned in each chapter of this dissertation, here we show other future work necessary to cover all the needs detected throughout the research work and that will continue to contribute to the difficulty in learning programming and address the other independent variables that may intervene in the process of teaching learning.

- **Controlled exercise recommendation:** We will study whether construction recommendations take into account the order in which subjects are taught to improve student learning. To this end, we will study recommendation systems based on content. These systems recommend elements 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 [71] model. This type of system reduces the size of the valuation matrix by transforming it into characteristics that represent common factors presented 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. Through EDM (Educational Data Mining), we can perform data mining to obtain patterns of academic collaboration, which will allow teachers to generate learning models based on the user profiles they detect in their students.[59]. For Romero et al. [67] EDM allows the use of data repositories in the analysis of students and their learning, with the aim of building computational approaches that combine data and theory to make practice work for the benefit of students.

- **SocialResponsibility:** We propose to continue with the extension of Scratch and CARAMBA to more public sector schools in the Milagro region, covering more variables that may affect the learning of children from 8 to 12 years old, to contribute to their development of computational thinking.

7.3 Epilogue

As a final conclusion, after reading these two famous phrases: "The mind is like a parachute ... It only works if we have it open..." *Albert Einstein.* and "True wisdom lies in recognizing one's own ignorance." *Socrates.*

We would like to paraphrase:

So we have to keep an open mind, accept that ignorance, despite so many years of hard work, is still present and that we have much to do.

Part V
Appendix

Appendix A

Questionnaires in spanish

In this appendix we provide the three questionnaires in Spanish that were asked to the university students.

A1. Edad: valor númerico

A2. Sexo: Femenino 0 Masculino 0

A3. Nivel de instrucción de los padres: Sin estudios - Primario - Secundario - Técnico - Superior/Postgrado

A4. Nivel socio-económico de la familia: Bajo - Medio - Alto

A5. Promedio de calificaciones en la secundaria:

A6. Promedio de calificaciones en el curso pre-universitario:

A7. Recibí clases de Fundamentos de Programación en el colegio SI NO

A8. Si tuviese la oportunidad de escoger mi opción de estudios, escogería la que estoy cursando actualmente

A9. Cumplo con regularidad las tareas encomendadas por el docente

A10. Asisto frecuentemente a clases

A11. Me interesa la asignatura Fundamentos de Programación

A12. Considero que tengo capacidad intelectual suficiente para superar la asignatura Fundamentos de Programación

A13. Considero que aprobaré la asignatura de Fundamentos de Programación

Figure A.1: Survey 1ST. Group homogeneity

B1. Tengo más posibilidades de ser promovido si se utiliza la herramienta pedagógica Scratch. Escala Linkert

B2. Considero que la herramienta Scratch contribuirá a mejorar mis calificaciones en la asignatura Fundamentos de Programación. Escala Linkert

B3. Considero que Scratch puede contribuir a las metodologías utilizadas por el docente de la asignatura de Fundamentos de Programación. Escala Linkert

B4. Fue necesario tener conocimientos previos de programación para poder usar la herramienta Scratch. Escala Linkert

B5. Me gustaría asistir a cursos donde se profundicen los conceptos y usos de la herramienta Scratch. Escala Linkert

B6. La inclusión de Scratch en el contenido de la asignatura de Fundamentos de Programación cambió positivamente mi perspectiva de la asignatura. Escala Linkert

B7. Me gustaría que el software Scratch sea considerado como herramienta pedagógica para impartir la clase de Fundamentos de Programación. Escala Linkert

B8. Creo que utilizar el Scratch frente a otros métodos de enseñanza puede mejorar las calificaciones de los estudiantes de la asignatura Fundamentos de Programación. Escala Linkert

B9. Creo que utilizar el Scratch frente a otros métodos de programación puede aumentar el interés y la motivación del alumnado por la asignatura Fundamentos de Programación. Escala Linkert

Figure A.2: Survey 2ND. Satisfaction with the Scratch tool

C1. Considero que he logrado aprendizajes sólidos y duraderos en la asignatura Fundamentos de Programación. Escala Linkert

C2. Ha sido fácil el aprendizaje de la asignatura. Escala Linkert

C3. Creo que lo que he aprendido es aplicable a otras asignaturas de mi carrera

C4. Estudié la asignatura con entusiasmo e interés. Escala Linkert

C5. Creo que lo que he aprendido lo aplicaré en mi futuro profesional. Escala Linkert

C6. Me siento motivado realizar fuera de clases actividades de profundización o complemento de los conocimientos adquiridos en esta asignatura. Escala Linkert

C7. Me siento motivado para terminar la carrera. Escala Linkert

C8. Obtuve buenas calificaciones en la asignatura. Escala Linkert

Figure A.3: Survey 3TH. Learning improvements

Appendix B

Tables and complementary figures

In this appendix, we present a group of tables and figures that were obtained in the investigation.

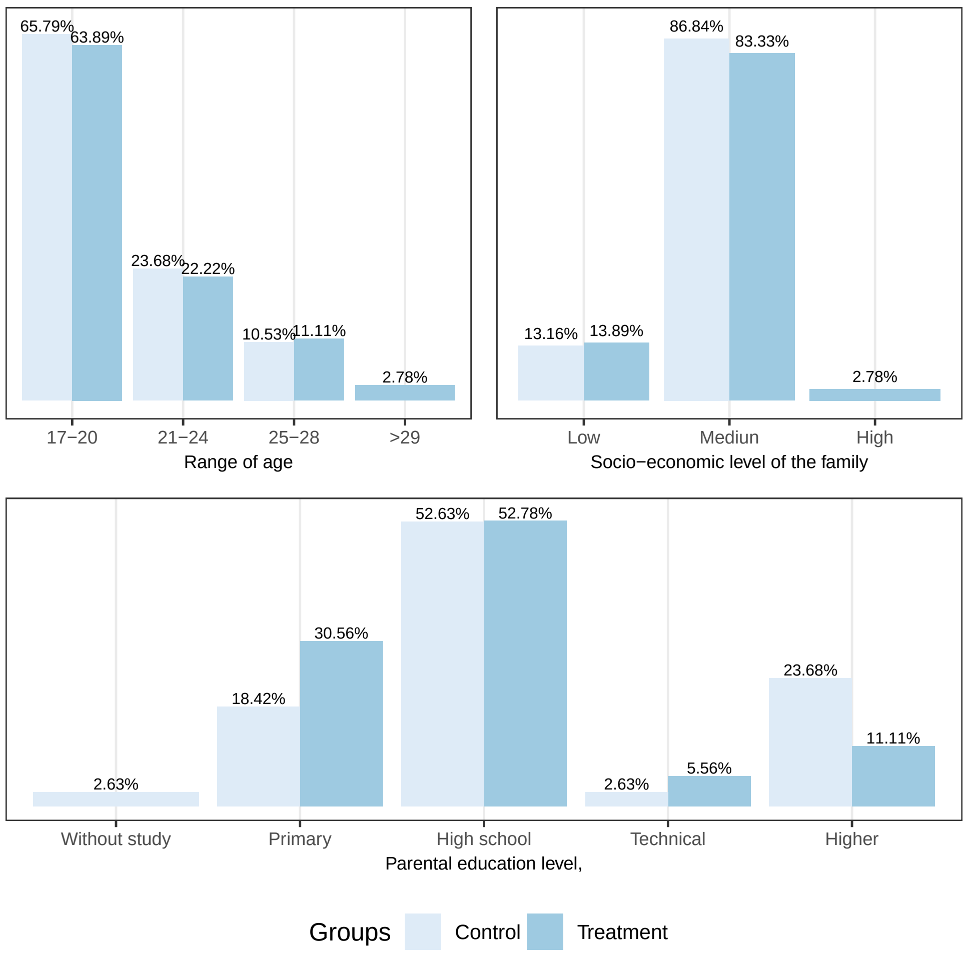


Figure B.1: Information of the student groups

		n	Avg	SD.	SE.	95% of the confidence interval for the mean		Min	Max
						Lower limit	Higher limit		
Age	Control group	38	20.35	2.635	.389	19.57	21.13	17	28
	Group using Scratch	36	21.09	3.387	.389	20.32	21.87	18	34
	Total	74	20.81	3.134	.284	20.25	21.37	17	34
Level of studies of the students parents	Control group	38	3.33	1.117	.165	2.99	3.66	1	5
	Group using Scratch	36	2.99	.931	.107	2.77	3.20	2	5
	Total	74	3.11	1.014	.092	2.93	3.30	1	5
Socio-economic level of students' families	Control group	37	1.87	.344	.051	1.76	1.97	1	2
	Group using Scratch	35	1.82	.420	.049	1.72	1.92	1	3
	Total	72	1.84	.392	.036	1.77	1.91	1	3
Average grades in high school	Control group	38	8.82	.544	.080	8.66	8.98	8.00	9.70
	Group using Scratch	36	8.87	.517	.059	8.75	8.99	7.79	10.00
	Total	74	8.85	.526	.047	8.76	8.95	7.79	10.00
Average grades in the pre-university course	Control group	32	8.53	1.616	.255	8.01	9.0506	7.30	18.00
	Group using Scratch	30	8.62	1.368	.164	8.29	8.9504	7.50	19.00
	Total	62	8.589	1.457	.139	8.31	8.86	7.30	19.00

Table B.1: Description of the sample object to study

		Sum of Squares	gl	Quadratic average	F	Sig
Age	Control group	15.874	1	15.874	1.624	.205
	Group using Scratch	1172.790	71	9.773		
	Total	1188.664	72			
Level of studies of the students parents	Control group	3.298	1	3.298	3.268	.073
	Group using Scratch	121.096	71	1.009		
	Total	124.393	72			
Socio-economic level of students family	Control group	.056	1	.056	.362	.549
	Group using Scratch	17.885	65	.154		
	Total	17.941	66			
Average grades in high school	Control group	.068	1	.068	.242	.623
	Group using Scratch	33.454	70	.279		
	Total	33.521	71			
Average grades in the pre-university course	Control group	.197	1	.197	.092	.763
	Group using Scratch	229.342	47	2.143		
	Total	229.538	48			

Table B.2: Analysis of variance (ANOVA) of a single factor

Appendix C

Application of Scratch & CARAMBA

In this appendix we present images of the application of Scratch & CARAMBA in a school context with the intervention of university students in the career of system engineering.

UNIVERSIDAD ESTATAL DE MILAGRO
UNEMI
Evolución Académica
UNIVERSIDAD DE SEVILLA
DIAGNOSTIC
Before initiating Phase I, a reasoning test was applied to the children

Figure C.1: Diagnostic test

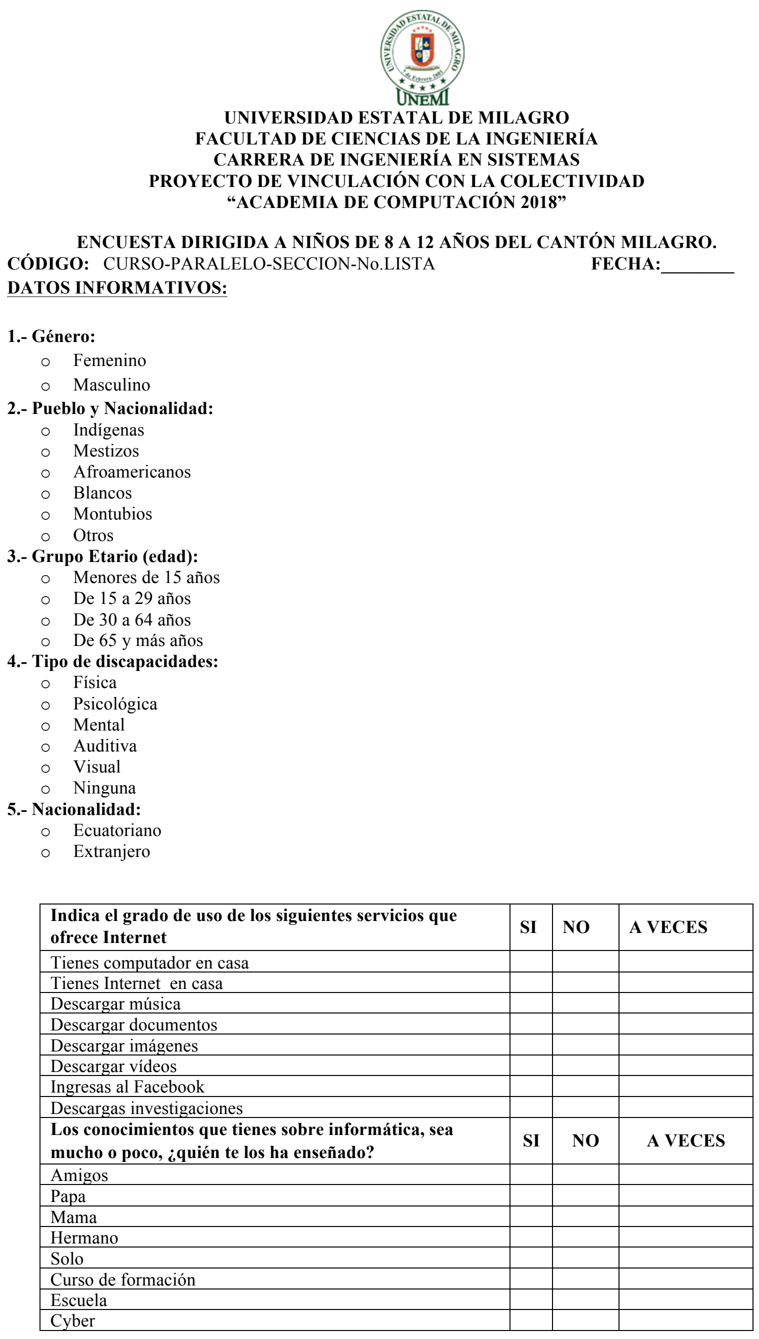
Figure C.2: Phase 1: teaching Scratch, guided mode

CARAMBA
PHASE II
Applying Scratch + CARAMBA
autonomous mode.
At the end the Test V2 is applied.

Figure C.3: Phase 2: applying Scratch + CARAMBA, autonomous mode

Appendix D
Computational thinking test

In this appendix, we present the test with which we evaluate the computational thinking, before and after our intervention, with Scratch



UNIVERSIDAD ESTATAL DE MILAGRO
FACULTAD DE CIENCIAS DE LA INGENIERÍA
CARRERA DE INGENIERÍA EN SISTEMAS
PROYECTO DE VINCULACIÓN CON LA COLECTIVIDAD
"ACADEMIA DE COMPUTACIÓN 2018"

ENCUESTA DIRIGIDA A NIÑOS DE 8 A 12 AÑOS DEL CANTÓN MILAGRO.

CÓDIGO: CURSO-PARALELO-SECCION-No.LISTA **FECHA:**________

DATOS INFORMATIVOS:

1.- Género:
- Femenino
- Masculino

2.- Pueblo y Nacionalidad:
- Indígenas
- Mestizos
- Afroamericanos
- Blancos
- Montubios
- Otros

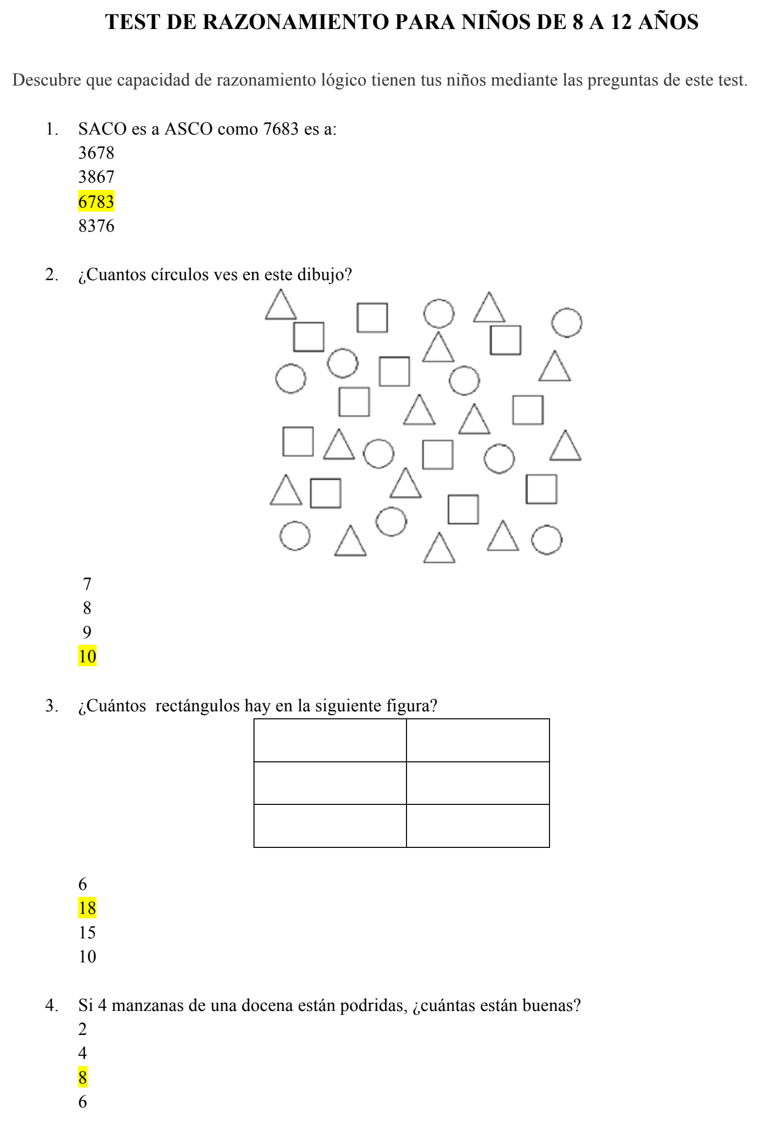
3.- Grupo Etario (edad):
- Menores de 15 años
- De 15 a 29 años
- De 30 a 64 años
- De 65 y más años

4.- Tipo de discapacidades:
- Física
- Psicológica
- Mental
- Auditiva
- Visual
- Ninguna

5.- Nacionalidad:
- Ecuatoriano
- Extranjero

Indica el grado de uso de los siguientes servicios que ofrece Internet	**SI**	**NO**	**A VECES**
Tienes computador en casa			
Tienes Internet en casa			
Descargar música			
Descargar documentos			
Descargar imágenes			
Descargar vídeos			
Ingresas al Facebook			
Descargas investigaciones			
Los conocimientos que tienes sobre informática, sea mucho o poco, ¿quién te los ha enseñado?	**SI**	**NO**	**A VECES**
Amigos			
Papa			
Mama			
Hermano			
Solo			
Curso de formación			
Escuela			
Cyber			

Figure D.1: Initial diagnostic test, informative data

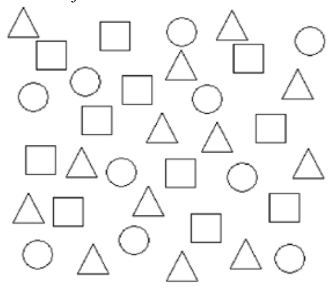


TEST DE RAZONAMIENTO PARA NIÑOS DE 8 A 12 AÑOS

Descubre que capacidad de razonamiento lógico tienen tus niños mediante las preguntas de este test.

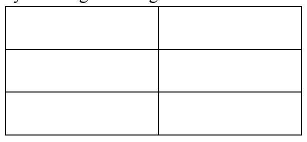
1. SACO es a ASCO como 7683 es a:
   3678
   3867
   [✓ 6783]
   8376

2. ¿Cuantos círculos ves en este dibujo?



   7
   8
   9
   [✓ 10]

3. ¿Cuántos rectángulos hay en la siguiente figura?

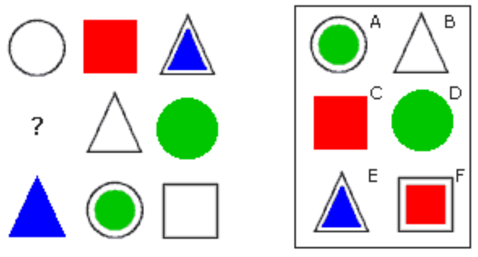


   6
   [✓ 18]
   15
   10

4. Si 4 manzanas de una docena están podridas, ¿cuántas están buenas?
   2
   4
   [✓ 8]
   6

Figure D.2: Initial reasoning test for children, part I

5. Busca entre las seis figuras de la derecha cuál es la que falta en el conjunto de la izquierda

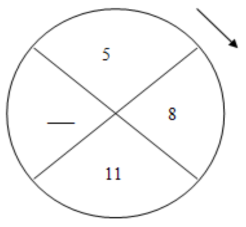


A
B
C
D
E
[✓ F]

6. ¿Cuál es el número que completa la serie?
6 - 12- 18- 24 - -36

34
28
[✓ 30]
32

7. ¿Cuál es el número que falta?



3
33
13
[✓ 14]

8. Observa la siguiente imagen, piensa y calcula el valor del cuadrado.

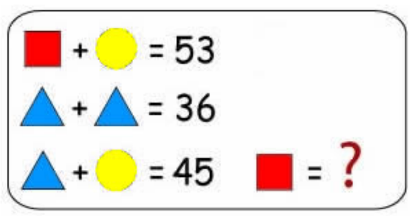


Figure D.3: Initial reasoning test for children, part II

24
26
28
30

9. Hoy he ido a comprar naranjas, la dependienta me ha dado 6, yo me he comido 1 y mi padre 2, otra se ha caído y se ha estropeado. ¿Cuántas naranjas me quedan?
2
3
4
Ninguna

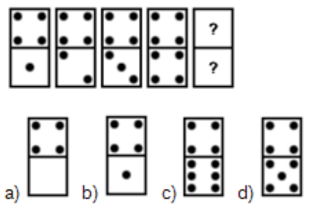
10. ¿Qué valor sigue en esta serie?
4 - 6 - 8 - 10 - ?

9
11
12
14

11. ¿Qué letra sigue en esta serie?
c - e - g - i - ?

k
a
l
j

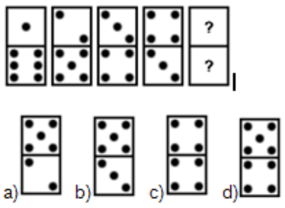
12. ¿Qué ficha sigue en la siguiente serie?



a
b
c
d

Figure D.4: Initial reasoning test for children, part III

13. ¿Qué ficha sigue en la siguiente serie?



a
b
c
d

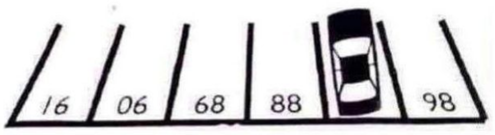
14. Si una mosca vive 5 días y en un día recorre 12 metros ¿cuánto recorrerá en 7 días?

60 metros
84 metros
77 metros

15. Si 7 gatos cazan 7 ratones en 7 minutos ¿cuantos minutos se demorará 1 gato en cazar 1 ratón?

7
1
5

16. ¿En qué número está estacionado de auto?



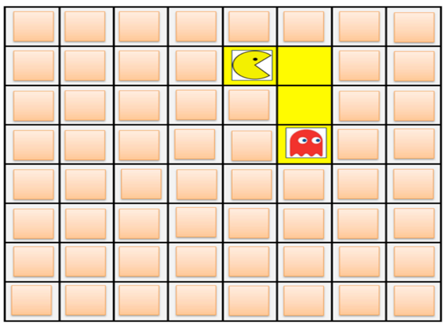
78
87
89
86

17. DIDIIDID es a 49499494 como DIIDIIDD es a:

94494499
49949944
49499494
94944949
49944949

Figure D.5: Initial reasoning test for children, part IV

DEPARTAMENTO DE EXTENSIÓN UNIVERSITARIA *Vinculación con la Colectividad*	**UNIVERSIDAD ESTATAL DE MILAGRO** UNEMI	
	Evaluación	20
	NOMBRE:	

¿Qué órdenes llevan a 'Pac-Man' hasta el fantasma por el camino señalado?	
	Opción A ➡ ➡ ⬇
	Opción B ➡ ⬇ ⬇
	Opción C ➡ ➡ ⬇ ⬇
	Opción D ⬇ ⬇ ➡

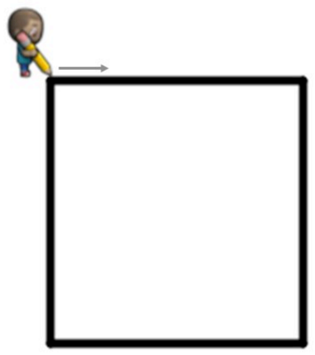
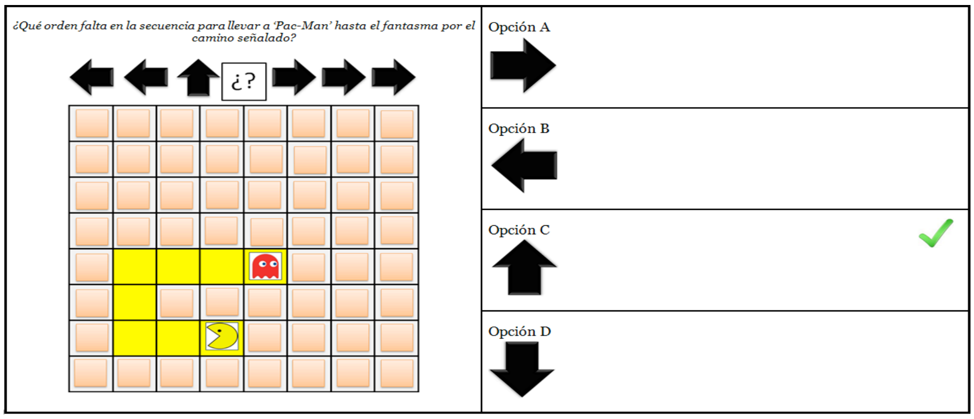
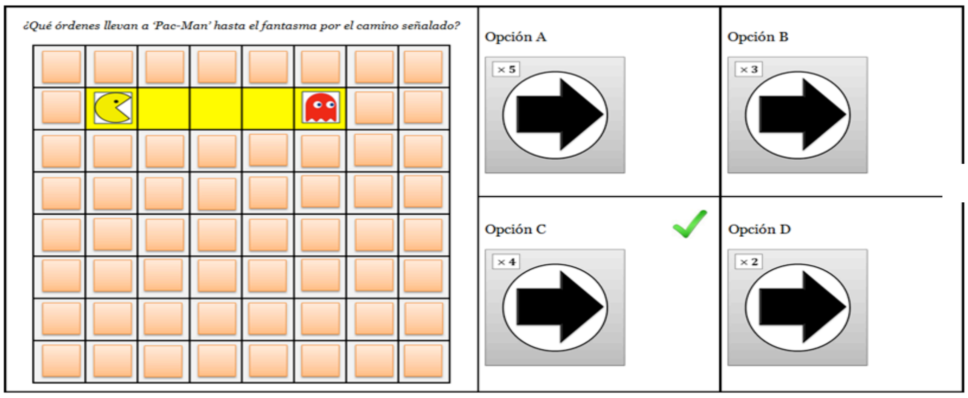
¿Qué órdenes debe ejecutar el artista para dibujar el cuadrado? Cada uno de los lados del cuadrado mide 100 píxeles.		
	Opción A mover hacia adelante 100 pixeles girar a la derecha por 90 grados mover hacia adelante 100 pixeles girar a la izquierda por 90 grados mover hacia adelante 100 pixeles girar a la derecha por 90 grados mover hacia adelante 100 pixeles	Opción B mover hacia adelante 25 pixeles girar a la derecha por 90 grados mover hacia adelante 25 pixeles girar a la izquierda por 90 grados mover hacia adelante 25 pixeles girar a la derecha por 90 grados mover hacia adelante 25 pixeles
	Opción C mover hacia adelante 50 pixeles girar a la derecha por 90 grados mover hacia adelante 50 pixeles girar a la derecha por 90 grados mover hacia adelante 50 pixeles girar a la derecha por 90 grados mover hacia adelante 50 pixeles	✓ Opción D mover hacia adelante 100 pixeles girar a la derecha por 90 grados mover hacia adelante 100 pixeles girar a la derecha por 90 grados mover hacia adelante 100 pixeles girar a la derecha por 90 grados mover hacia adelante 100 pixeles

Figure D.6: Computational thinking test for children, part I


¿Qué orden falta en la secuencia para llevar a 'Pac-Man' hasta el fantasma por el camino señalado?
¿?
Opción A
Opción B
Opción C
Opción D


¿Qué órdenes llevan a 'Pac-Man' hasta el fantasma por el camino señalado?
Opción A
× 5
Opción B
× 3
Opción C
× 4
Opción D
× 2

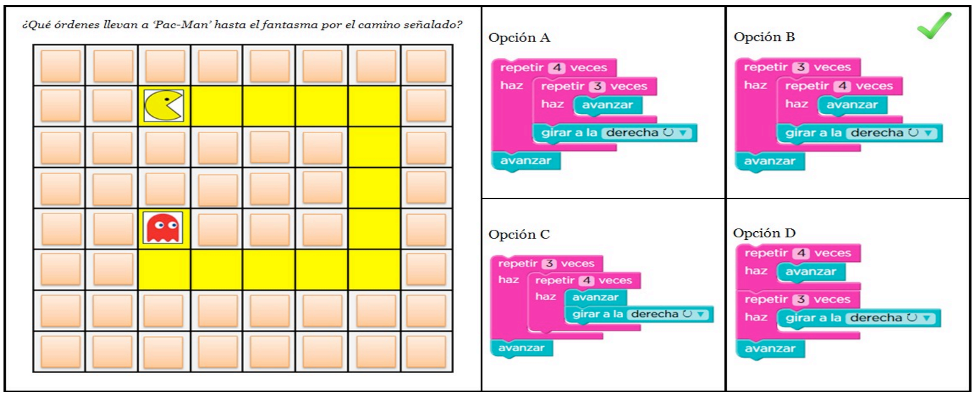
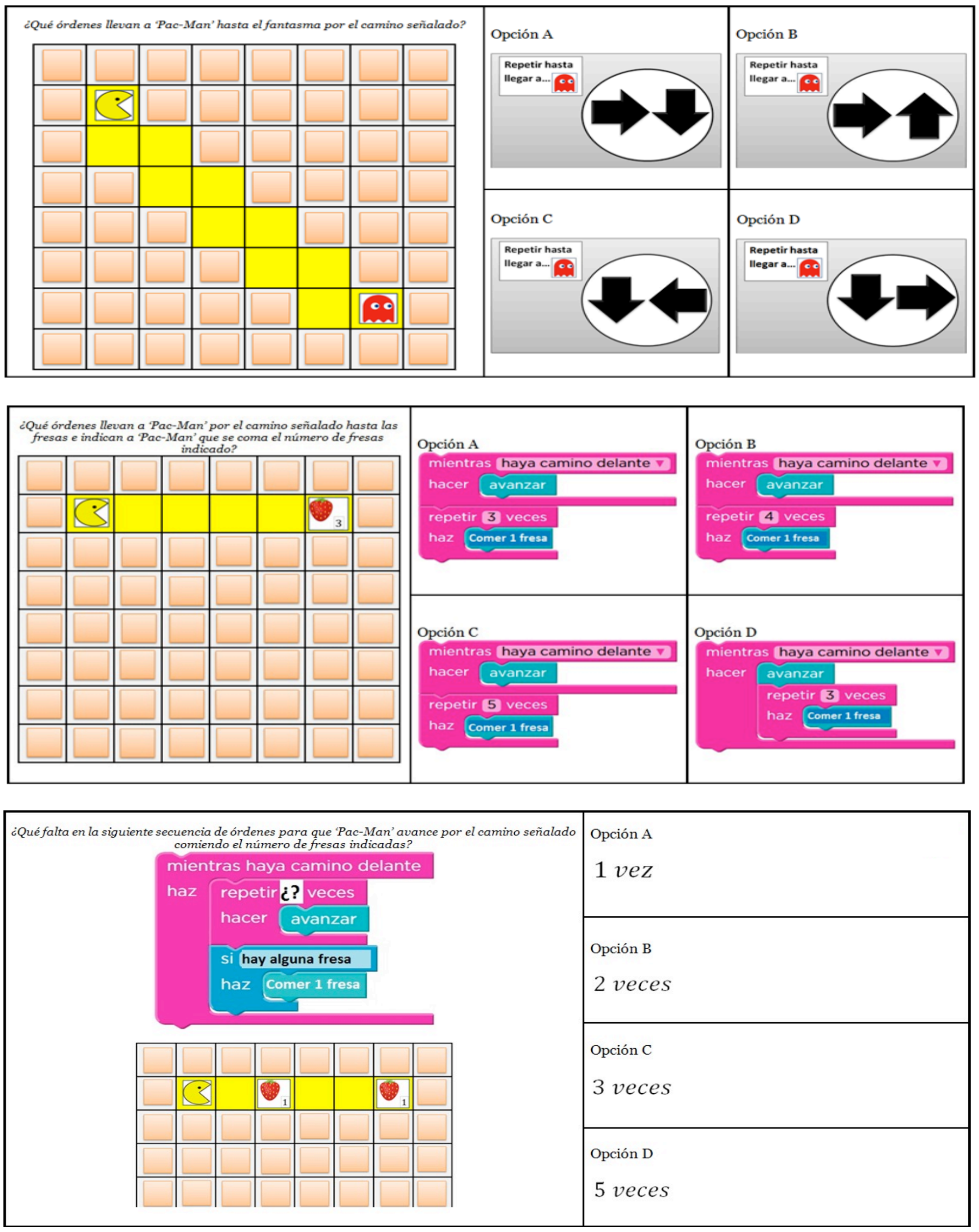


Figure D.7: Computational thinking test for children, part II



¿Qué órdenes llevan a 'Pac-Man' hasta el fantasma por el camino señalado?

Opción A

Repetir hasta llegar a...

Opción B

Repetir hasta llegar a...

Opción C

Repetir hasta llegar a...

Opción D

Repetir hasta llegar a...

¿Qué órdenes llevan a 'Pac-Man' por el camino señalado hasta las fresas e indican a 'Pac-Man' que se coma el número de fresas indicado?

Opción A

```
mientras haya camino delante ▾
hacer avanzar
repetir 3 veces
haz Comer 1 fresa
```

Opción B

```
mientras haya camino delante ▾
hacer avanzar
repetir 4 veces
haz Comer 1 fresa
```

Opción C

```
mientras haya camino delante ▾
hacer avanzar
repetir 5 veces
haz Comer 1 fresa
```

Opción D

```
mientras haya camino delante ▾
hacer avanzar
      repetir 3 veces
      haz Comer 1 fresa
```

¿Qué falta en la siguiente secuencia de órdenes para que 'Pac-Man' avance por el camino señalado comiendo el número de fresas indicadas?

```
mientras haya camino delante
haz  repetir ¿? veces
     hacer avanzar
     si hay alguna fresa
     haz Comer 1 fresa
```

Opción A

1 *vez*

Opción B

2 *veces*

Opción C

3 *veces*

Opción D

5 *veces*

Figure D.8: Computational thinking test for children, part III

Bibliography

[1] F. I. Anfurrutia, A. Álvarez, M. Larrañaga, and J.-M. López-Gil. Visual programming environments for object oriented programming: Acceptance and effects in students' motivation. *VAEP-RITA*, 5(1):11–18, 2017.

[2] A. Aquino-Leal and D. James-Ferreira. Learning programming patterns using games. *International Journal of Information and Communication Technology Education (IJICTE)*, 12(2):23–34, 2016.

[3] M. Armoni, O. Meerbaum-Salant, and M. Ben-Ari. From scratch to "real" programming. *Trans. Comput. Educ.*, 14(4):25: 1–25: 15, feb 2015. ISSN 1946-6226.

[4] J. J. Arnett. The neglected 95%: why american psychology needs to become less american. *American Psychologist*, 63(7):602, 2008.

[5] L. Barker. Snap, create, and share with scratch (case study 5). In *Promising Practices workshop*. NCWIT (National Center for Women and Information Technology), 2008.

[6] D. Bau, J. Gray, C. Kelleher, J. Sheldon, and F. Turbak. Learnable Programming: Blocks and Beyond. *arXiv preprint arXiv:1705.09413*, 2017. ISSN 00010782.

[7] J. Beltrán, H. Sánchez, and M. Ricob. Quantitative and qualitative analysis of learning programming at the central university of ecuador. *Revista Tecnológica ESPOL*, 28(5):194–210, 2015.

[8] N. C. Brown and A. Altadmri. What's new in bluej 4: Git, stride and more. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 734–734, 2017.

[9] N. Bubica and I. Boljat. Strategies for Teaching Programming to Meet New Challenges : State of the Art. *Ciet*, (June):1–6, 2014.

[10] S.-y. Byun, J. L. Meece, and M. J. Irvin. Rural-nonrural disparities in post-secondary educational attainment revisited. *American educational research journal*, 49(3):412–437, 2012.

[11] S.-y. Byun, J. L. Meece, and C. A. Agger. Predictors of college attendance patterns of rural youth. *Research in Higher Education*, 58(8):817–842, 2017.

[12] R. Cano González. Tutoría universitaria y aprendizaje por competencias. ¿cómo lograrlo?. *REIFOP*, 12(1):181–204, 2009.

[13] 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*, pages 255–271. Springer International Publishing, Cham, 2018. ISBN 978-3-319-60937-9.

[14] M. E. Caspersen and J. Bennedsen. Instructional design of a programming course: A learning theoretic approach. In *Proceedings of the Third International Workshop on Computing Education Research*, ICER '07, pages 111–122, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-841-1.

[15] S. Cooper, W. Dann, and R. Pausch. Alice: A 3-d tool for introductory programming concepts. *Journal of Computing Sciences in Colleges.*, pages 107–116, 2000.

[16] E. Costelloe. Teaching programming the state of the art. *The Center for Research in IT in Education. Dept. of Computer Science Education. Dublin: Trinity College*, 2004.

[17] J. Cárdenas-Cobo, D. Benavides, M. D'Armas Regnault, M. Vinueza, and J. Rodas. Programación con la herramienta scratch +caramba. una experiencia de aprendizaje significativo. *LACCEI*, pages 107–144, 2017.

[18] J. Cárdenas-Cobo, A. Puris, P. Novoa-Hernández, J. A. Galindo, and D. Benavides. Recommender systems and scratch: An integrated approach for enhancing computer programming learning. *IEEE Transactions on Learning Technologies*, pages 1–1, 2019. ISSN 1939-1382.

[19] T. R. da Silva, T. Medeiros, H. Medeiros, R. Lopes, and E. Aranha. Ensino-aprendizagem de programação: uma revisão sistemática da literatura. *Revista Brasileira de Informática na Educação*, 23(01):182, 2015. ISSN 1414-5685.

[20] C. Desrosiers and G. Karypis. A Comprehensive Survey of Neighborhood-based Recommendation Methods. In K. P. e. Ricci F.,

Rokach L., Shapira B., editor, *Recommender Systems Handbook*, chapter Chapter 4, pages 107–144. Springer, Boston, MA, 2011. ISBN 9780387858203.

[21] F. Díaz and G. Hernández. *Estrategias docentes para un aprendizaje significativo*, volume 2. México: McGraw-Hill, 2002.

[22] H. Y. Durak and T. Güyer. Design and development of an instructional program for teaching programming processes to gifted students using scratch. In *Curriculum development for gifted education programs*, pages 61–99. IGI Global, 2018.

[23] A. Escribano González. Aprendizaje cooperativo y autónomo en la enseñanza universitaria. *Universidad de Salamanca*, 1995.

[24] C. Feng and M. Chen. The effects of goal specificity and scaffolding on programming performance and self-regulation in game design. *British Educational Research Association*, 39(1):285–302, 2013.

[25] T. Ferrer-Mico, M. Àngel Prats-Fernàndez, and A. Redo-Sanchez. Impact of scratch programming on students' understanding of their own learning process. *Procedia - Social and Behavioral Sciences*, 46:1219 – 1223, 2012. ISSN 1877-0428. 4th World Conference on Educational Sciences (WCES-2012) 02-05 February 2012 Barcelona, Spain.

[26] C. L. Fraile. Estudio y trabajo autónomos del estudiante. *Métodos y modalidades de enseñanza centradas en el desarrollo de competencias. Madrid: Alianza Universidad*, pages 191–223, 2006.

[27] I. Friss de Kereki and A. Adorjan. Innovación en la Enseñanza Inicial de la Programación. In *Innovando en Educación Superior: Experiencias clave en Latinoamérica y el Caribe 2016-2017*, volume 3, chapter Integración de TIC's, pages 55–66. Universidad de Chile, Santiago, Chile, 2017. ISBN 978-956-19-1015-7.

[28] R. Fry. Latinos in higher education: Many enroll, too few graduate. *Pew Hispanic Center*, 2002.

[29] R. Garcia, K. Falkner, and R. Vivian. Systematic literature review: Self-regulated learning strategies using e-learning tools for computer science. *Computers & Education*, 123:150–163, 2018.

[30] S. Georgantaki and S. Retails. Using Educational Tools for Teaching Object Oriented Design and Programming. *Journal of Information Technology Education*, 7(2):111–130, 2007.

[31] A. Gǿmes and A. Méndez. An environment to improve programming education. international conference on computer systems and technologies-compsystech '07 proceedings. In *07*, volume 40, page 88. ACM New York, NY, USA, 2007.

[32] M. Guerrero, D. S. Guamán, and J. C. Caiza. Review of support tools in the teaching-learning process of programming. *Revista Politécnica*, 35(1), 2015.

[33] J. Henrich, S. J. Heine, and A. Norenzayan. Most people are not weird. *Nature*, 466(7302):29, 2010.

[34] I. Hernán, J. Á. Velázquez, and C. A. Lázaro. Dos herramientas educativas para el aprendizaje de programación: generación de comentarios y creación de objetos. *VII Congreso Internacional de Interacci{ó}n Persona-Ordenador*, (July 2014):325–334, 2006.

[35] N. Hollender, C. Hofmann, M. Deneke, and B. Schmitz. Integrating cognitive load theory and concepts of human–computer interaction. *Computers in human behavior*, 26(6):1278–1288, 2010.

[36] W. Huitt. Bloom et al.'s taxonomy of the cognitive domain. *Educational psychology interactive*, 22, 2004.

[37] S. Ida and S. Ketil. *Technology research explained*, chapter Technology research explained. Technical report, pages 6–10. SINTEF ICT, NORWAY, 2007.

[38] E. D. Jaramillo. Incidencia de la implementación del ambiente de programación scratch, en los estudiantes de media técnica, para el desarrollo de la competencia laboral general de tipo intelectual exigida por el ministerio de educación nacional colombiano. Master's thesis, Universidad Autónoma de Bucaramanga, Colombia, 2013.

[39] M. Jonas and M. Sabin. Computational thinking in greenfoot: Ai game strageties for cs1: Conference workshop. *J. Comput. Sci. Coll.*, 30(6):8–10, June 2015. ISSN 1937-4771.

[40] D. Jones. A weird view of human nature skews psychologists' studies, 2010.

[41] F. Kalelioğlu. A new way of teaching programming skills to k-12 students: Code. org. *Computers in Human Behavior*, 52:200–210, 2015.

[42] B. Kaucic and T. Asic. Improving introductory programming with Scratch? *2011 Proceedings of the 34th International Convention MIPRO*, (January 2011):1095–1100, 2011.

[43] C. Kelleher and R. Pausch. Lowering the barriers to programming. *ACM Computing Surveys*, 37(2):83–137, 2005. ISSN 03600300.

[44] H. Keuning, J. Jeuring, and B. Heeren. A systematic literature review of automated feedback generation for programming exercises. *ACM Trans. Comput. Educ.*, 19(1):3:1–3:43, Sept. 2018. ISSN 1946-6226.

[45] M. Koorsse, C. Cilliers, and A. Calitz. Programming assistance tools to support the learning of it programming in south african secondary schools. *Computers & Education*, 82:162–178, 2015.

[46] D. A. Kranch. Teaching the novice programmer: A study of instructional sequences and perception. *Education and Information Technologies*, 17 (3):291–313, 2012. ISSN 13602357.

[47] W. M. Kunkle. The Impact of Different Teaching Approaches and Languages on Student Learning of Introductory Programming Concepts. *ProQuest Dissertations and Theses*, 16(September):175, 2010. ISSN 19466226.

[48] M. Kölling. The greenfoot programming environment. *ACM Transactions on Computing Education*, 10–4(17):16–20, 2010.

[49] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen. A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3):14, 2005. ISSN 00978418.

[50] V. Lee and K. Frank. Student characteristics that facilitate transfer from two-year to four-year colleges. *Sociology of Education*, 63:178–193, 1990.

[51] C. López-Escribano and R. Sánchez-Montoya. Scratch y Necesidades Educativas Especiales: Programación para todos. *RED Revista de Educación a Distancia*, 34:2–14, 2012. ISSN 14684462.

[52] C. López-Escribano and R. Sánchez-Montoya. Scratch y necesidades educativas especiales: Programación para todos. *RED Revista de Educación a Distancia.*, (34):2–14, 2012.

[53] 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*, volume 39 of *SIGCSE '07*, pages 223–227, New York, New York, USA, 2007. ACM Press. ISBN 1595933611.

[54] J. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk. Programming by choice: urban youth learning programming with scratch. *SIGCSE '08 Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 367–371, 2008. ISSN 0097-8418.

[55] B. L. McCombs and J. S. Whisler. The role of affective variables in autonomous learning. *Educational Psychologist*, 24(3):277–306, 1989.

[56] R. P. Medeiros, G. L. Ramalho, and T. P. Falcão. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, pages 1–14, aug 2019. ISSN 0018-9359.

[57] A. Mesoudi, K. Magid, and D. Hussain. How do people become w.e.i.r.d.? migration reveals the cultural transmission mechanisms underlying variation in psychological processes. *PLOS ONE*, 11(1):1–17, 01 2016.

[58] N. Moroni and P. Señas. Un entorno para el aprendizaje de la programación. In *II Congreso Argentino de Ciencias de la Computación, 410. Ateneo de Profesores Universitarios de Computación y Sistemas*, 1996.

[59] E. Nankani, S. Simoff, S. Denize, and L. Young. Supporting strategic decision making in an enterprise university through detecting patterns of academic collaboration. In *International United Information Systems Conference*, pages 496–507. Springer, 2009.

[60] M. Niess. Preparing teachers to teach science and mathematics with technology: Developing a technology pedagogical content knowledge. *Teaching and Teacher Education*, 21(5):509 – 523, 2005. ISSN 0742-051X.

[61] S. Papert. *Mindstorms: Children, Computers, and Powerful Ideas.*, chapter 1. Basic Books., Inc. New York, USA, 1980.

[62] S. M. Pope, J. Fagot, A. Meguerditchian, D. A. Washburn, and W. D. Hopkins. Enhanced cognitive flexibility in the seminomadic himba. *Journal of Cross-Cultural Psychology*, 50(1):47–62, 2019.

[63] A. Radenski. "python first": A lab-based digital introduction to computer science. iticse '06. In *11th Annual Conference on Innovation and Technology in Computer Science Education Bologna*, Italy, 2006.

[64] F. Rauner and Maclean. Handbook of technical and vocational education and research in training. *Springer*, 2018.

[65] M. Resnick. Sowing the crat for a more creative society. learning & leading with technology. *ISTE (International Society for Technology in Education).*, 2008.

[66] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, et al. Scratch: Programming for all. *Communications of the ACM.*, 52(11):60–67, 2009.

[67] C. Romero and S. Ventura. Educational data mining: A review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(6):601–618, Nov 2010. ISSN 1094-6977.

[68] M. Román González. *Codigoalfabetización y pensamiento computacional en Educación Primaria y Secundaria: validación de un instrumento y evaluación de programas.* Tesis, Universidad Nacional de Educación a Distancia (España). Escuela Internacional de Doctorado. Programa de Doctorado en Educación, May 2016.

[69] K. Ruggeri, L. Bojanic, H. Bokhorst, L.and Jarke, and O. N. S. Marev, a. S.and Ojinaga-Alfageme. Editorial: Advancing methods for psychological assessment across borders. *Frontiers in psychology*, 1(10):503, 2019.

[70] R. H. Sampieri, C. Collado, Fernández, P. B. Lucio, and M. L. Pérez. *Metodología de la investigación*, volume 6. Mcgraw-hill México, 1998.

[71] 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*, pages 285–295. ACM, 2001.

[72] E. Segredo, G. Miranda, and C. León. Hacia la educación del futuro: El pensamiento computacional como mecanismo de aprendizaje generativo. *Education in the Knowledge Society (EKS)*, 18(2):33, 2017. ISSN 2444-8729.

[73] S. Shuhidan, M. Hamilton, and D. D'Souza. A taxonomic study of novice programming summative assessment. In *Proceedings of the Eleventh Australasian Conference on Computing Education - Volume 95*, ACE '09, pages 147–156, Darlinghurst, Australia, Australia, 2009. Australian Computer Society, Inc. ISBN 978-1-920682-76-7.

[74] T. Y. Sim and S. L. Lau. Online tools to support novice programming: A systematic review. In *2018 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, pages 91–96, Nov 2018.

[75] D. Teague, C. Fidge, and Y. Xu. Combining unsupervised and invigilated assessment of introductory programming. In *Proceedings of the*

Australasian Computer Science Week Multiconference, ACSW '16, pages 9:1–9:10, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4042-7.

[76] I. Utting, M. Cooper, S.and K¨Olling, J. Maloney, and M. Resnick. Alice, greenfoot, and scratch – a discussion. *ACM Transactions on Computing Education*, 10–4(17):16–20, 2010.

[77] A. Vihavainen, M. Paksula, and M. Luukkainen. Extreme apprenticeship method in teaching programming for beginners. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 93–98. ACM, 2011.

[78] J. L. Whalley, R. Lister, E. Thompson, T. Clear, P. Robbins, P. Ajith Kumar, and C. Prasad. An australasian study of reading and comprehension skills in novice programmers, using the bloom and solo taxonomies. In *Conferences in Research and Practice in Information Technology Series*, 2006.

[79] J. Willem, D. Alderliesten, A. Guijt, F. Doolaard, L. Stegman, and J. Tilro. Identifying characteristics of block-based programming languages supporting children in learning robotics programming. pages 1–18.

[80] J. M. Wing. Computational Thinking: What and Why? *thelink - The Magazine of the Varnegie Mellon University School of Computer Science*, 49(3):1–6, 2010.

[81] 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*, volume 41, page 2, New York, New York, USA, 2009. ACM Press. ISBN 9781605581835.

[82] H. Yildiz Durak. The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving. *Technology, Knowledge and Learning*, Aug 2018. ISSN 2211-1670.

[83] I. Yoon, J. Kim, and W. Lee. The analysis and application of an educational programming language rur-ple for a pre-introductory computer science course. *Cluster Computing*, 19(1):529–546, 2016. ISSN 1573-7543.

This document was typeset on 2020/2/25 using RC–BOOK α2.11 for $\LaTeX_{2\epsilon}$.
Should you want to use this document class, please send mail to
contact@tdg-seville.info.