Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking

Dr. Scratch: Análisis Automático de Proyectos Scratch para Evaluar y Fomentar el Pensamiento Computacional

Jesús Moreno-León
Programamos.es. Spain.
jesus.moreno@programamos.es

Gregorio Robles
Universidad Rey Juan Carlos. Spain.
grex@gsyc.urjc.es

Marcos Román-González
Universidad Nacional de Educación a Distancia (UNED). Spain.
mroman@edu.uned.es

Abstract
One of the barriers to entry of computer programming in schools is the lack of tools that support educators in the assessment of student projects. In order to amend this situation this paper presents Dr. Scratch, a web application that allows teachers and students to automatically analyze projects coded in Scratch, the most used programming language in primary and secondary education worldwide, to check if they have been properly programmed, learn from their mistakes and get feedback to improve their code and develop their Computational Thinking (CT) skills. One of the goals of Dr. Scratch, besides supporting teachers in the evaluation tasks, is to act as a stimulus to encourage students to keep on improving their programming skills. Aiming to check its effectiveness regarding this objective, workshops with students in the range from 10 to 14 years were run in 8 schools, in which over 100 learners analyzed one of their Scratch projects with Dr. Scratch, read the information displayed as feedback by Dr. Scratch, and tried to improve their projects using the guidelines and tips offered by the tool. Our results show that at the end of the workshop, students increased their CT score and, consequently, improved their coding skills.

Keywords
Computational thinking, learning, coding, Scratch, assessment

Resumen
Una de las barreras de entrada de la programación informática en las escuelas es la falta de herramientas que ayuden al profesorado en la evaluación de los proyectos del alumnado. Con el objetivo de resolver esta situación, este artículo presenta Dr. Scratch, una aplicación web que permite a educadores y alumnos analizar automáticamente proyectos Scratch, el lenguaje de programación más utilizado globalmente en educación primaria y secundaria, para comprobar si se han programado correctamente, aprender de sus errores y recibir retroalimentación para mejorar su código y desarrollar el Pensamiento Computacional (PC). Uno de los objetivos de Dr. Scratch, además de ayudar al docente en las tareas de evaluación, es ser un estímulo para animar a los aprendices a seguir mejorando sus habilidades de programación. Para comprobar la efectividad de la herramienta en este sentido, se organizaron talleres en 8 colegios con alumnos de entre 10 y 14 años en los que los estudiantes analizaron uno de sus proyectos Scratch con Dr. Scratch, leyeron la información del informe de resultados e intentaron mejorar sus proyectos usando los consejos ofrecidos por la herramienta. Al finalizar el taller los alumnos mejoraron su puntuación de PC así como sus habilidades como programadores.

Palabras clave
Pensamiento computacional, aprendizaje, programación, Scratch, evaluación

Introduction

In the last decade we have witnessed a resurgence of programming and Computational Thinking (Wing, 2006) (CT) in schools (Lye & Koh, 2014). The educational use of coding, which had been introduced in the 70s and 80s mainly with the Logo programming language (Papert & Solomon, 1971), has come back strong due to new visual programming languages, like Alice, Kodu and especially Scratch, which allow young students to program applications without the need to learn the complex syntax of traditional programming languages.

Scratch (Resnick et al., 2009) is a visual programming environment designed for children over 6 years old, which also offers a website where users can share their projects and exchange ideas or suggestions with other (young) programmers. Scratch is massively used all over the world, with more than seven million registered users and more than ten million shared projects in the repository[1]. One of the main goals of Scratch is that programming becomes an educational tool to develop other skills and to improve learning of other disciplines (Resnick, 2013). As a result, Scratch is being used both in extracurricular activities (Kafai, Fields, & Burke, 2012) and in all levels of formal educational environments, both in schools (Moreno-León & Robles, 2015), high schools (Meerbaum-Salant, Armoni, & Ben-Ari, 2013) and even universities (Malan & Leitner, 2007) worldwide.

Nevertheless, there is a lack of tools that support educators when evaluating student programs and to assess the development of CT. This situation is partly caused by the fact that there is a lack of agreement in a definition of the CT concept, and in the way it should be included in the curriculum (Grover & Pea, 2013). This paper presents Dr. Scratch, a free/open source web tool that analyzes Scratch projects to (1) offer feedback to educators and learners and (2) assign a CT score to the projects. Learners can use this feedback to improve their programs, but also can realize how to improve their programming abilities. To test the effectiveness of Dr. Scratch, we have set up a set of workshops to measure the impact of its use on learning. Results show positive results and hint areas of future development.

The paper is structured as follows: the Background section reviews different proposals and tools that try to assist educators in the assessment of the CT of students; then the features included in Dr. Scratch are explained; the approach followed in preparing the workshops to test Dr. Scratch with programming learners is detailed in the Methodology section; the results of the workshops, both quantitative and qualitative, are shown in the Findings section; finally, in the Conclusions we summarize our study, discuss the limitations of the tool and present some new features the development team is working on.

Background

There is lack of tools that support educators in the assessment of the development of CT

[1] See http://scratch.mit.edu/statistics/

and the evaluation of projects programmed by students. Regarding the Scratch programming language, several authors have proposed different approaches to evaluate the development of CT of learners by analyzing their projects, but most of these approaches are based exclusively on a manual analysis.

Wilson, Hainey, and Connolly (2012) suggest a scheme to gauge the level of programming competence demonstrated by a student by analyzing a project in terms of programming concepts (such as threads, conditional statements or variables), code organization (variable names, sprite names and extraneous blocks) and designing for usability (like functionality, instructions or originality, among others).

In this line, Seiter and Foreman (2013) developed the school Progression of Early Computational Thinking Model, a framework to assess CT in primary students programming with Scratch by synthesizing "measurable evidence from student work with broader, more abstract coding design patterns, which are then mapped onto computational thinking concepts" (Seiter & Foreman, 2013, p. 59)

In the paper *"New frameworks for studying and assessing the development of computational thinking"*, Brennan and Resnick (2012) introduced a strategy based on project portfolio analysis using a visualization tool called Scrape (Wolz, Hallberg, & Taylor, 2011), which seems to be no longer available, although their proposal is completed with artifact-based interviews and design scenarios.

Aiming to assist educators with a tool that could be used to partly automate the assessment of Scratch projects, Boe et al. (2013) developed Hairball, a static code analyzer that detects potential issues in the programs, such as code that is never executed, messages that no object receives or attributes not correctly initialized. After a two-week Scratch-based summer camp, Hairball was used to assess Computer Science learning in terms of event-driven programming, initialization of state and message passing (Franklin et al., 2013).

The Hairball architecture, based on plug-ins, is ideal to add new features. In a previous work, the authors developed two plug-ins to detect two bad programming habits we frequently detect in our work as instructors with high school students (Moreno & Robles, 2014):

- *convention.SpriteNaming*[2] analyzes a Scratch project to check if the names of the sprites begin with the string *Sprite*, which indicates that the programmer has not modified the default name that Scratch assigns to an object. It should be noted that while using the default name for sprites produces no error in the program if its implementation is correct, it makes the readibility of the program more difficult, especially when the number of sprites is high (i.e., more than ten).
- *duplicate.DuplicateScripts*[3] analyzes a Scratch project to find duplicate scripts, which are repeated programs within a project. For such type of structures, Scratch custom blocks should be used.

In order to check if these bad programming habits are also common in the projects

[2] https://github.com/jemole/hairball/blob/master/hairball/plugins/convention.py
[3] https://github.com/jemole/hairball/blob/master/hairball/plugins/duplicate.py

shared in the Scratch website, we randomly downloaded and analyzed 100 projects, detecting that 79% of the inspected projects presented not personalized object names, while 62% included repeated code (Moreno & Robles, 2014). These figures encouraged us to develop a tool to help both learners and educators to detect issues in the code to improve their programming skills.

The fact that Hairball is executed from the command-line, as it is based on *Python* scripts that the evaluator has to manually launch, makes it not suitable for many educators that are not confident with such an environment, let alone for young students. For this reason, we decided to create a web-based service, Dr. Scratch, that allows the analysis of Scratch projects easily.

Introducing Dr. Scratch

Dr. Scratch[4] is a free/open-source web application that allows to easily analyze Scratch projects using Hairball plug-ins, as well as to obtain feedback that can be used to improve programming skills and develop CT. To analyze a project with Dr. Scratch an .sb or an .sb2 file can be uploaded, as the tool supports both 1.4 and 2.0 Scratch versions, or the users can directly copy the URL of the project. The ability to analyze projects from the URL has been implemented using *getsb2*[5].

When a Scratch project is analyzed, Dr. Scratch informs the user of the degree of development of CT demonstrated in that project, assigning a CT score. Being based on Hairball, Dr. Scratch detects certain bad habits of programming or potential errors, such as non-significant sprite names, repetition of code, code that is never executed and the incorrect initialization of object attributes.

In order to assign the CT Score, Dr. Scratch infers the competence demonstrated by the developer on the following seven concepts: abstraction and problem decomposition, logical thinking, synchronization, parallelism, algorithmic notions of flow control, user interactivity and data representation. The evaluation of the competence level of each of these concepts follows the rules in Table 1, which was designed based on the proposals presented in the Background section by remixing some of their ideas with the support of educators from different educational levels who use Scratch in their classrooms.

Depending on the CT score, which may range from 0 to 21 points, distinct data is displayed in the results page. Thus, if the CT level is low it is assumed that the user is a novice programmer and, consequently, the tool will only show basic information of the most important improvements to perform in the code. As the score increases, Dr. Scratch will show more information of the analyzed projects. Thus, advanced users receive a feedback report with all available information both in terms of CT skills and bad programming habits. Figures 1 and 2 illustrate the differences in the quantity and complexity of the information displayed on screen depending on the CT Score.

[4] http://drscratch.org/
[5] https://github.com/nathan/getsb2

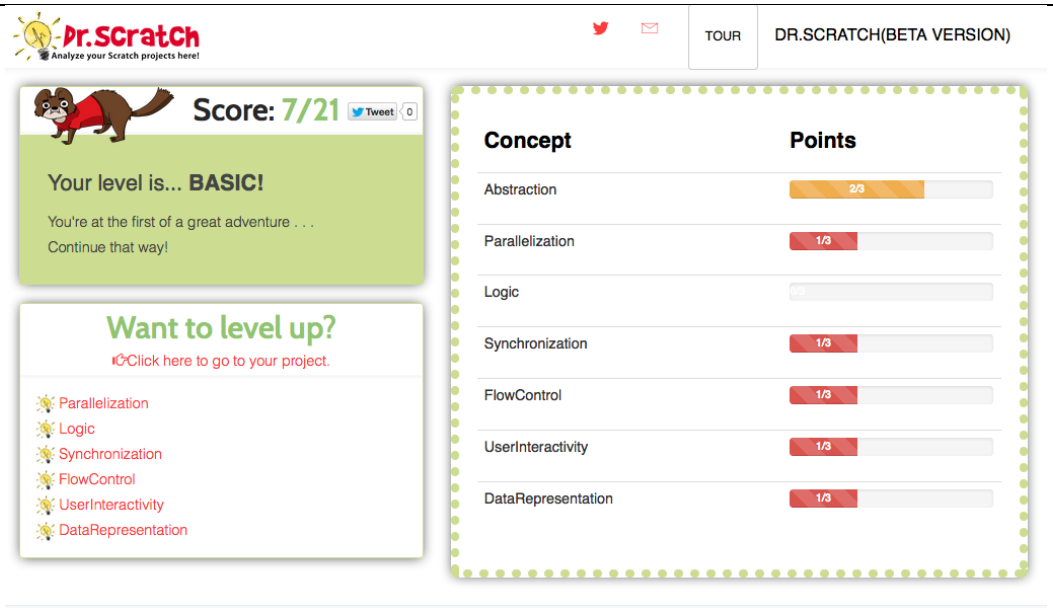

Fig. 1. Dr. Scratch analysis results for a project with basic CT Score

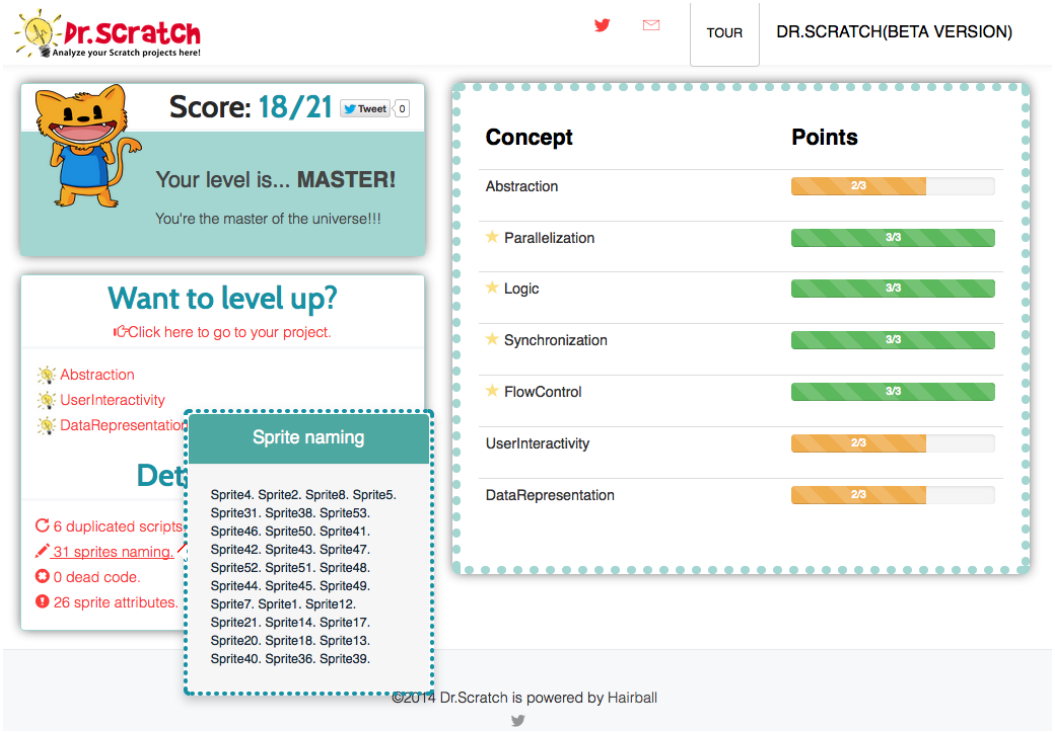

Fig. 2. Dr. Scratch analysis results for a project with advanced CT Score

Figure 3 can be used to illustrate the operation of the CT assessment. Thus, following the rules in Table 1, the first script of the picture would be cataloged as *basic* in terms of data representation, as it modifies some of the object attributes (position and orientation). The second script, however, would be considered to demonstrate a *developing* level, because a variable is utilized. Finally, the third script would prove a *proficient* level on this concept, as an operation on lists is performed.

In those aspects where there is room for improvement, the tool provides links to

information that can be used to improve. For example, if a project has been awarded with one point in parallelism, Dr. Scratch provides a link to sample source code and an explanation of how to perform several actions at the same time in a program (see Figure 4).

CT Concept	Competence Level			
	Null (0)	**Basic (1 point)**	**Developing (2 points)**	**Proficiency (3 points)**
Abstraction and problem decomposition	-	More than one script and more than one sprite	Definition of blocks	Use of clones
Parallelism	-	Two scripts on green flag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, create clone, two scripts when %s is > %s, two scripts on when backdrop change to
Logical thinking	-	If	If else	Logic operations
Synchronization	-	Wait	Broadcast, when I receive message, stop all, stop program, stop programs sprite	Wait until, when backdrop change to, broadcast and wait
Flow control	-	Sequence of blocks	Repeat, forever	Repeat until
User Interactivity	-	Green flag	Key pressed, sprite clicked, ask and wait, mouse blocks	When %s is >%s, video, audio
Data representation	-	Modifiers of sprites properties	Operations on variables	Operations on lists

Table 1. Competence Level for each CT concept.

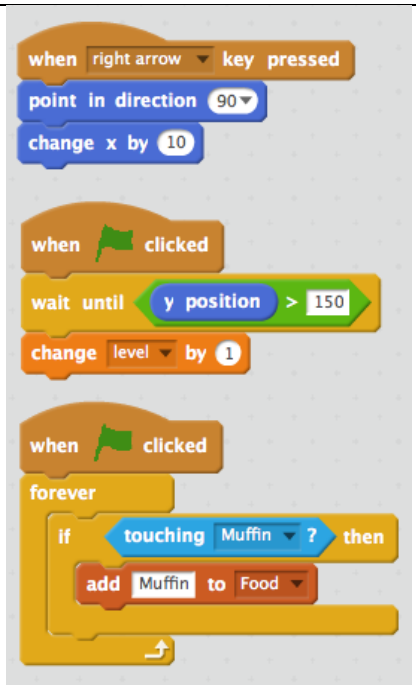

Fig. 3. Different competence levels of data representation: *basic* (top), *developing* (center) and *proficient* (bottom).

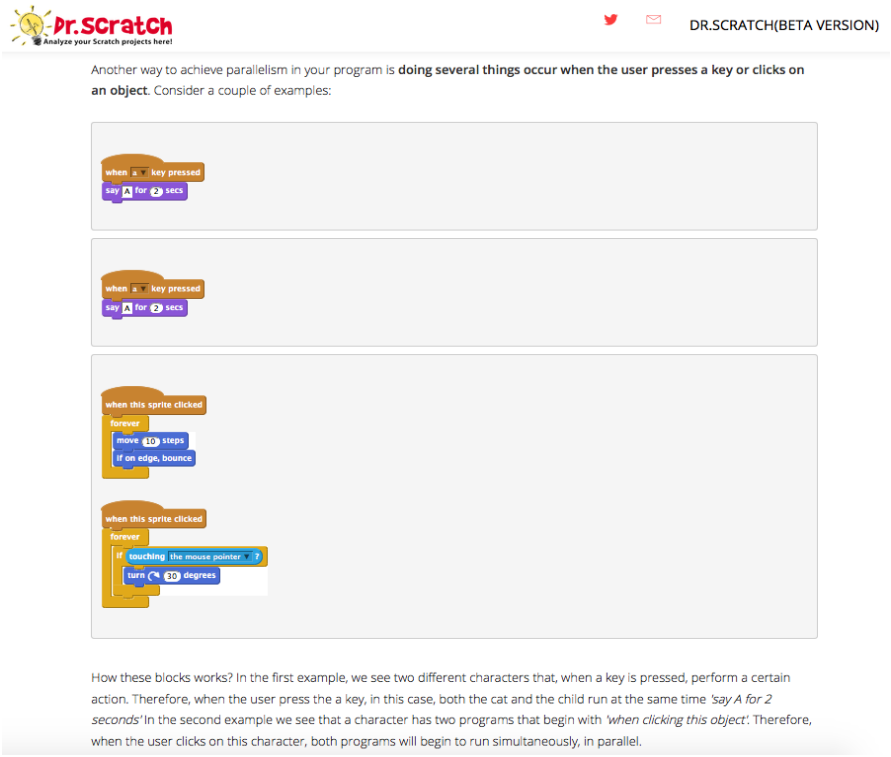

Fig. 4. Ideas and tips provided by Dr. Scratch to improve projects by incorporating parallelism.

Methodology

In order to assess the effectiveness of Dr. Scratch as a tool to assist programming learners, we performed a number of workshops with students between 10 and 14 years from eight Spanish primary and secondary schools (see Figure 5). These students had previously learned to program with Scratch for several weeks in their schools.

During the one-hour workshop, students were given a questionnaire with some questions they had to answer while performing different tasks. The tasks and questions were as follows:

1) Visit the Dr. Scratch website:
   a) What do you think about the website? Do you find it attractive?
   b) After reading the information on the website, what do you think Dr. Scratch can be used for?
2) Analyze one of your Scratch projects with Dr. Scratch.
   a) Is it easy to analyze projects with Dr. Scratch?
   b) What was your score?
   c) According to Dr. Scratch, what is the CT level for that score?
   d) How did you feel when you saw the results?
   e) Why?
3) From the results page, after analyzing a project, click on some of the links to receive information that could help you improve your code.
   a) Write the title of the page you clicked on.
   b) Do you understand the information in the results page?
   c) After reading the information, do you feel like trying something new?
4) Using the information that appeared in the help page you selected, try to improve your project by adding something new.
   a) Are the ideas and tips in the results page enough to improve your program?
   b) After performing some modifications, analyze again your project with Dr. Scratch. What is the new score?
5) Do you have any other comments?

Fig. 5. Dr. Scratch workshop at Lope de Vega Primary School, Madrid.

Characteristics of the study sample

Table 2 shows the characteristics of the study sample regarding the age of the participants, formed by a group of 109 students between 10 and 14 years. The mean age of the sample was 11.50, while the median was 11 and the mode was 10.

N	**Valid**	109
	Missing	0
Mean		11.50
Median		11.00
Mode		10
Standard Deviation		1.392
Variance		1.937
Minimum		10
Maximum		14

Table 2. Age of students participating in the investigation

Figure 6 shows the percentage of participating students for each age group. As can be seen, a majority of participants were 10 or 11 years old.

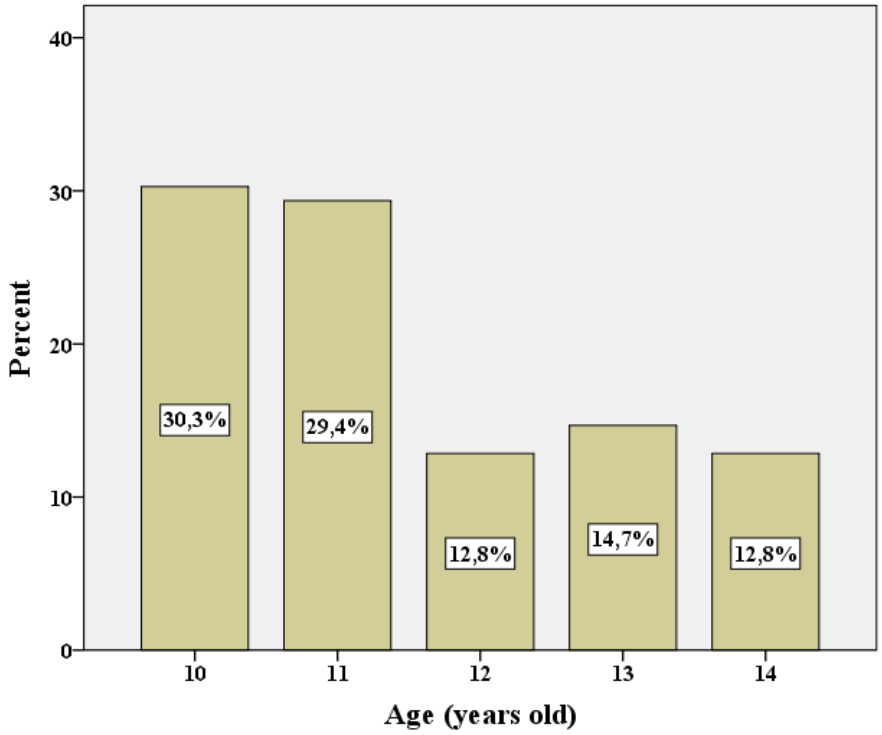

Figure 6. Percentage of students by age group (in years old)

In terms of gender, Table 3 shows the percentage of boys and girls participating in the investigation; 57.8% of participants were boys while 42.2% were girls. This difference can be explained by the fact that in some of the schools the experiment was carried out in non-compulsory subjects, such as ICT or technology, where there usually is a majority of boys. On the other hand, the gender of the students was not recorded in one of the participating schools, which explains the 19 missing records on Table 3.

Although there has been significant research into gender issues in computer science education (Beyer, Rynes, Perrault, Hay & Haller, 2003) and programming (Carter & Jenkins, 1999), that topic is out of the scope of this study. Nonetheless, future research on if boys and girls react to and/or learn different with Dr. Scratch is planned.

		Frequency	Percentage	Valid Percentage	Cumulative Percentage
Valid	**Boy**	52	47.7	57.8	57.8
	Girl	38	34.9	42.2	100.0
	Total	90	82.6	100.0	
Missing	**System**	19	17.4		
Total		109	100.0		

Table 3. Percentage of students by gender

Finally, regarding the educational stage of the students, as shown in Table 4, a majority of participants was enrolled in Primary Education, although we also had a significant group of students from Secondary Education.

		Frequency	Percentage	Valid Percentage	Cumulative Percentage
Valid	**Primary Education**	75	68.8	68.8	68.8
	Secondary Education	34	31.2	31.2	100.0
	Total	109	100.0	100.0	

Table 4. Percentage of students by educational stage

Findings

Figures 7, 8, 9, 10 and 11 show the answers of the students to some of the questions of the questionnaire. According to the responses, a majority of the students found the Dr. Scratch website attractive, as can be seen in Figure 7, and most of students believed that analyzing projects with Dr. Scratch was easy (Figure 8). Regarding their feelings after analyzing their projects, shown in Figure 9, a majority of learners felt good when they saw the CT score, although 3% of the respondents indicated that they felt bad. In regard to the information displayed by Dr. Scratch, Figure 10 shows that most of the students were able to understand it; however, 5.6% of the learners answered that they did not understand the information obtained. Finally, being one of the goals of Dr. Scratch to stimulate self-learning by offering a gamifyied environment, we were interested in the response of students after obtaining feedback. In this sense, Figure 11 shows that Dr. Scratch boosts the willingness to improve programming skills.

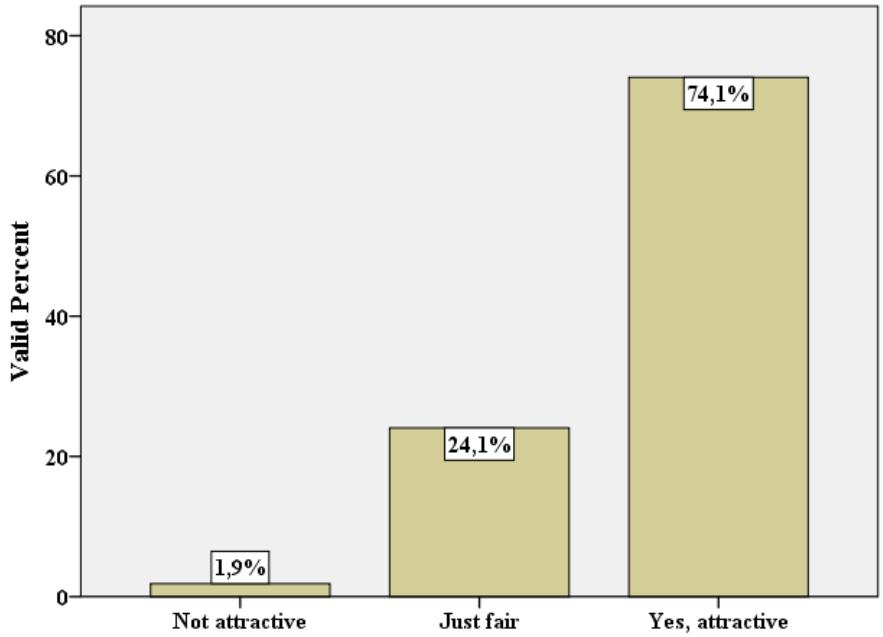

Figure 7. What do you think about the website? Do you find it attractive?

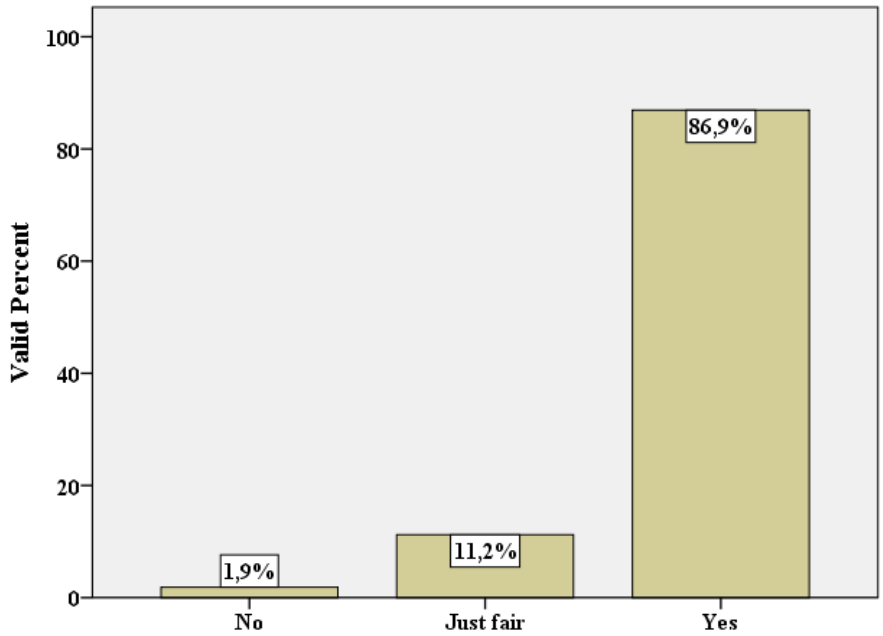

Figure 8. Analyze one of your Scratch projects with Dr. Scratch. Is it easy to analyze projects with Dr. Scratch?

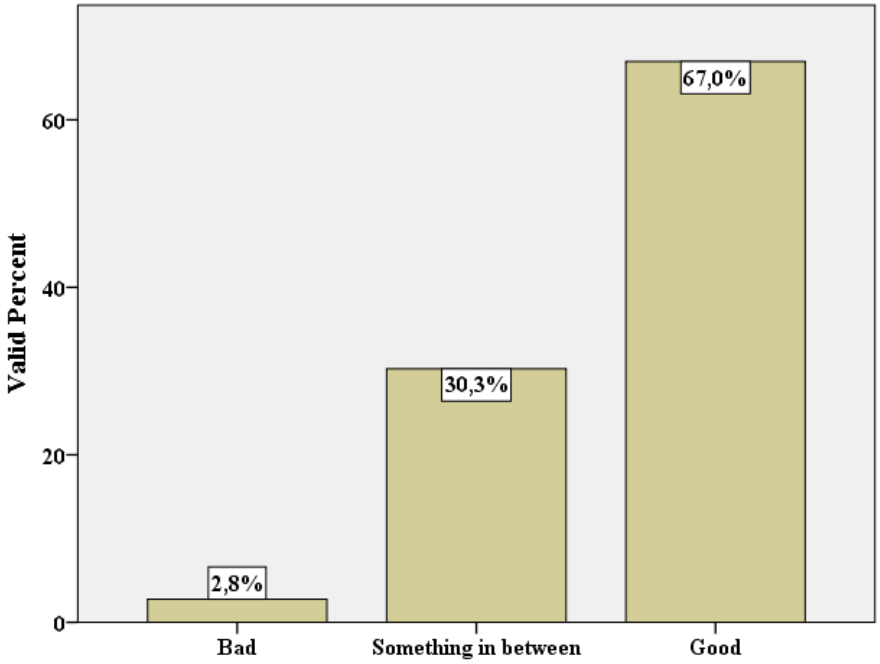

Figure 9. How did you feel when you saw the results?

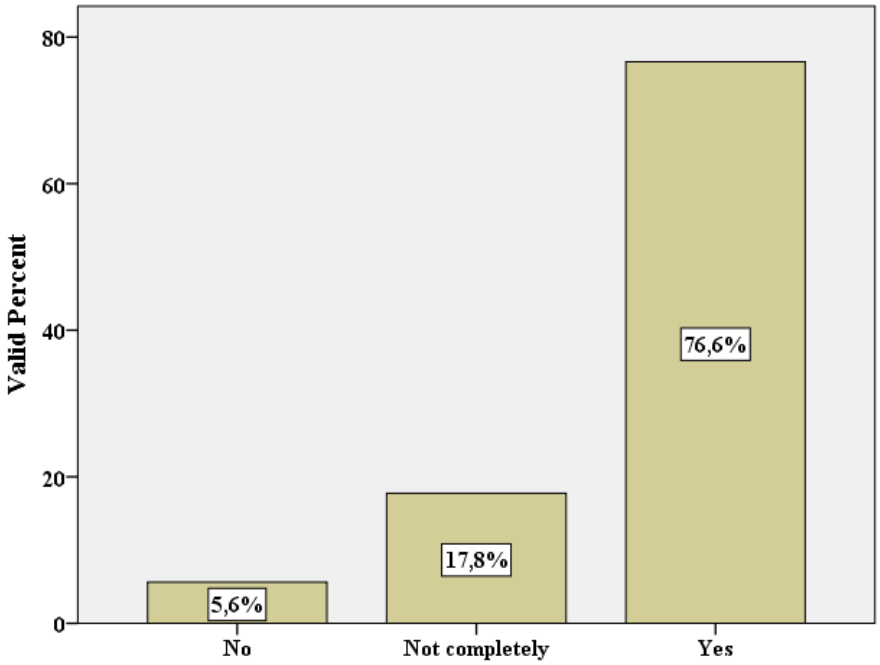

Figure 10. Do you understand the information in the results page?

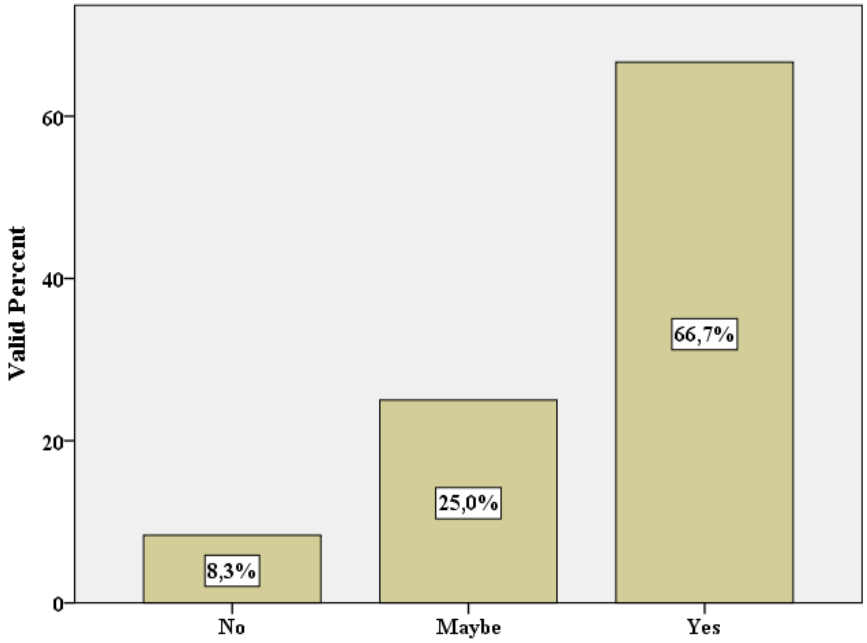

Figure 11. After reading the information, do you feel like trying something new?

Table 5, Figures 12 and 13 show the results of the analysis of the Scratch projects developed by the students before and after reading the feedback reports by Dr. Scratch. As it can be seen, there was an increase in the results, as the mean of the pre-test analysis is 11.82, while the mean of the post analysis is 13.52.

		Before Dr. Scratch Feedback (Pre-test)	***After Dr. Scratch Feedback (Post-test)***
Mean		11,82	13.52
Median		12	14
Mode		[11, 12, 15]	16
Std. Deviation		3.093	3.257
Skewness		.028	-.171
Minimum		5	5
Maximum		20	21
Percentiles	**10**	8	9
	20	9	10.40
	30	10	11
	40	11	13
	50	12	14
	60	13	15
	70	14	16
	80	15	16
	90	15.60	17

Table 5. Results of the projects' analysis before and after reading feedback by Dr. Scratch.

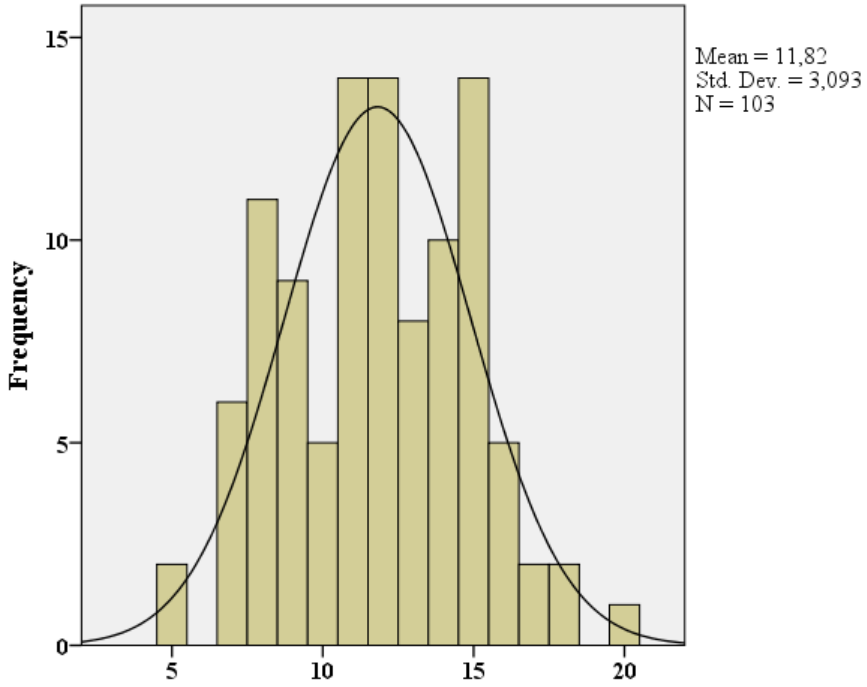

Figure 12. What was your score? (Pre-test scores)

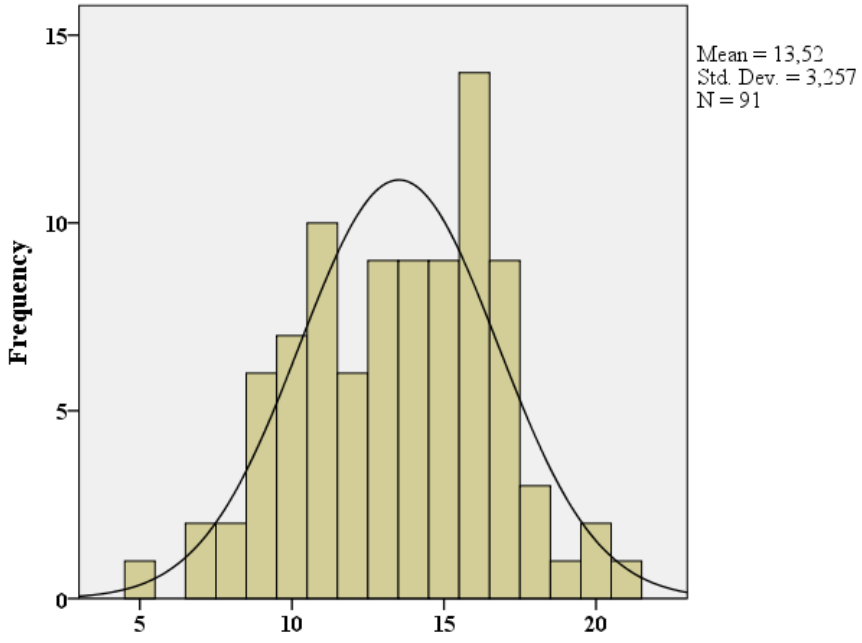

Figure 13. After performing some modifications, analyze again your project with Dr. Scratch. What is the new score? (Post-test scores)

To be able to prove that the improvement experienced by the students was statistically significant, we performed a *t*-test for paired samples, establishing a 95% confidence level ($\alpha = 0.05$) for our statistical decisions.

Table 6 shows the statistics of the 88 students who correctly indicated the pre and post analysis results of their projects. There were 21 not completed records out of the 109 participating students, because pre-test or post-test scores were not correctly specified by students. Several circumstances apply here: from students who specify just their CT level (low, medium, high) instead of their CT score, to Internet connection problems or an error in Dr. Scratch. For the 88 complete records, the mean value of CT score increased from 12.00 in the pre-test to 13.45 in the post-test.

Paired Samples Statistics					
		Mean	**N**	**Std. Deviation**	**Std. Error Mean**
Pair 1	**Pre-test Score**	12.00	88	2.983	.318
	Post-test Score	13.45	88	3.216	.343

Table 6. Statistics of paired data

Paired Samples Test									
		Paired Differences					**t**	**df**	**Sig. (2-tailed)**
		Mean	**Std. Deviation**	**Std. Error Mean**	**95% Confidence Interval of the Difference**				
					Lower	**Upper**			
Pair 1	**Pretest Score - Postest Score**	-1.455	1.523	.162	-1.777	-1.132	-8.959	87	**.000**

Table 7. *t*-test for paired data in the pre-test and post-test

The results of the *t*-test for paired data are shown in Table 7. As p(t) = 0.000 << 0.05, the null hypothesis of equality of means is rejected and we can therefore state that there are significant differences between the pre-test and post-test, which indicates that the use of Dr. Scratch helped the participating students develop their CT.

Aiming to assess the real impact of using Dr. Scratch on the development of CT, the effect size of the experiment is considered to be a good indicator as it is independent of sample size, and can be calculated according to the following formula (Cohen, 1990):

$$d = \frac{|\bar{X}pre - \bar{X}post|}{\sqrt{\frac{s_{pre}^2 + s_{post}^2}{2}}}$$

For our investigation the effect size was 0.47, which is considered as a moderate effect (Ellis, 2010). Nevertheless it must be noted that this effect was experienced after just 1 hour of treatment, consisting in the workshop, which highlights the impact that Dr. Scratch had on the learners and draws attention to its potential as a tool to foster CT by using it as a supporting tool in a programming course.

As expected, there is a positive and very significant correlation between pre-test and post-test scores, which is shown in Table 8.

Paired Samples Correlations				
		N	Correlation	Sig.
Pair 1	Pretest Score * Postest Score	88	.882	.000

Table 8. Correlations of paired data

Figure 14 shows the scatter plot for the sample when comparing the pre-test and post-test scores. As can be seen, all cases fall into the improvement area, which means that the scores in the post-test were equal or bigger than the pre-test scores for all of the 88 students; therefore, none of the learners decreased his/her score during the experiment.

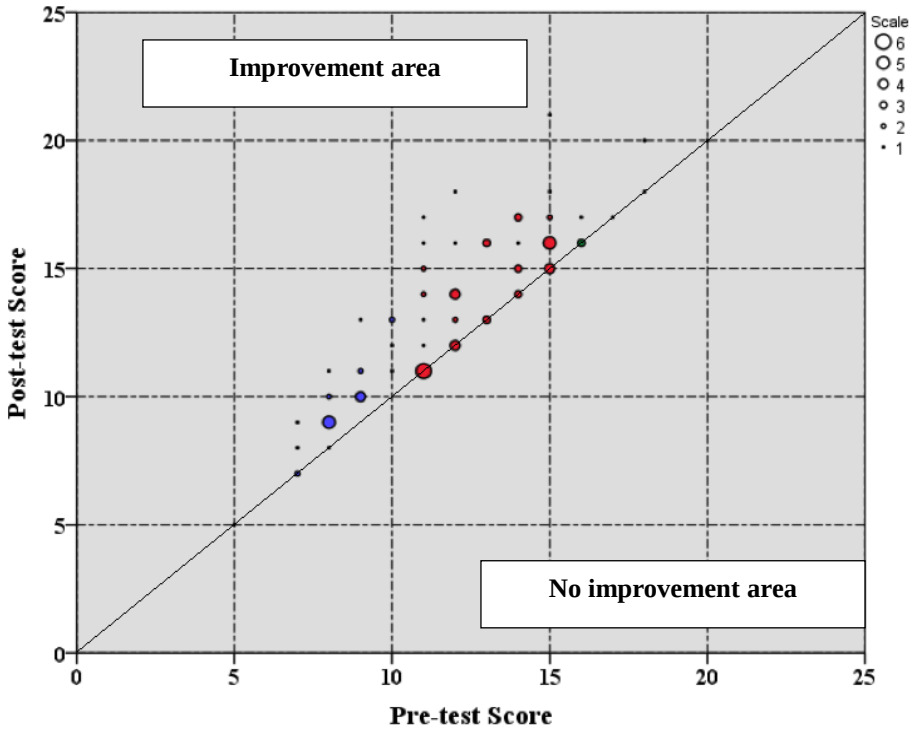

Figure 14. Scatter plot comparing post-test and pre-tests scores. Cases when the post-test score is equal or bigger than the pre-test score fall into the improvement area. Cases when the post-test score is less than pre-test score fall into the no improvement area.

We performed several significance analysis of the differences of pretest-postest depending on several factors. In the first place, the significance analysis was performed based on the initial score. Thus, we divided the sample in three sub-samples depending on the pre-test score:

- Sub-sample "Low initial score": pre-test scores ≤ 10 (blue color in Figure 14)
- Sub-sample "Medium initial score": pre-test scores ≤ 15 (red color in Figure 14)
- Sub-sample "High initial score": pre-test scores ≤ 21 (green color in Figure 14)

Table 9 shows the significance and correlation values both for the total sample and for each sub-sample. The results indicate that the use of Dr. Scratch generated significant improvements in the development of CT in the "medium initial score" sub-sample (11-15) and in the "low initial score" sub-sample (0-10), although the improvement is slightly lower in the latter case. However, Dr. Scratch did not generate a significant

improvement in the "high initial score" sub-sample (16-21). These results are, to some degree, in line with what was expected, as they indicate that students with basic and medium initial levels, where there is more room for improvement, are able to make use of the information provided by Dr. Scratch to enhance their score in just one hour, while the tool seem less useful for those students with a high initial score where improvements are harder to achieve. The differences between the initial and medium levels were also relatively expected, as novice learners have to struggle with the difficulties of the early steps in Scratch and the feedback provided by the tool. Nonetheless, Dr. Scratch offers CT-dependent feedback as in first tests with learners the authors noted that too much information in early phases were counterproductive. Future research will help us modify and adapt the feedback reports in this direction.

	N	**Post-Pre difference**	**t**	**p (t)**	**Significant difference 95% confidence?**	**r**	**p (r)**
Full sample	88	1.455	8.959	0.000 < 0.05	Yes	0.88	0.000
Sub-sample Low IL	25	1.440	6.896	0.000 < 0.05	Yes	0.89	0.000
Sub-sample Medium IL	55	1.618	6.951	0.000 < 0.05	Yes	0.61	0.000
Sub-sample High IL	8	0.375	1.426	0.197 > 0.05	No	0.90	0.002

Table 9. Significance and correlation values depending on pre-test score

Aiming to check if the use of Dr. Scratch had a different effect on primary and secondary students, we performed the significance analysis on the differences of pretest-posttest based on the educational stage. Table 10 shows the significance, effect size and correlation values, both for the total sample and for each sub-sample (Primary and Secondary education). As can be seen, both in primary and secondary education the use of Dr. Scratch generated a significant improvement, although the improvement was bigger in the latter stage.

	N	**Post-pre difference**	**t**	**p (t)**	**Significant difference 95% confidence?**	**r**	**p (r)**	**d**	**Effect Size**
Full sample	88	1.455	8.959	0.000 < 0.05	Yes	0.88	0.000	0.47	Moderate
Sub-sample Primary	64	1.234	7.294	0.000 < 0.05	Yes	0.90	0.000	0.40	Moderate (-)
Sub-sample Secondary	24	2.042	5.540	0.000 < 0.05	Yes	0.87	0.000	0.60	Moderate (+)

Table 10. Significance and correlation values depending on education stage

The analysis of covariance, shown in Table 11, confirms the significant effect of the educational stage on the post-test controlling the baseline differences in the pre-test, as p ($F_{\text{Educational Stage}}$) = 0.03 < 0.05

Tests of Between-Subjects Effects					
Dependent Variable: Post-test Score					
Source	**Type III Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
Corrected Model	710.667[a]	2	355.333	159.678	.000
Interception	22.806	1	22.806	10.248	.002
Pre-test Score	70.541	1	708.541	318.401	.000
Educational Stage	10.801	1	10.801	4.854	**.030**
Error	189.151	85	2.225		
Total	16830.000	88			
Corrected Total	899.818	87			
a. R Squared = .790 (Adjusted R Squared = .785)					

Table 11. Analysis of covariance. Dependent variable: post-test score.

Figure 15 can be used to illustrate the bigger improvement of secondary students. As can be seen, there is a bigger density of cases with the same pretest-posttest score (students do not improve, although they do not decline either after using Dr. Scratch) in Primary Education, while in Secondary Education the bigger density is translated to the improvement area. These results encourage us to plan future research in order to determine whether the differences between primary and secondary students are due to our tool, because of the language used and the examples included in the feedback report, or are related to the maturation meta-cognitive development that, in consequence, improves with age.

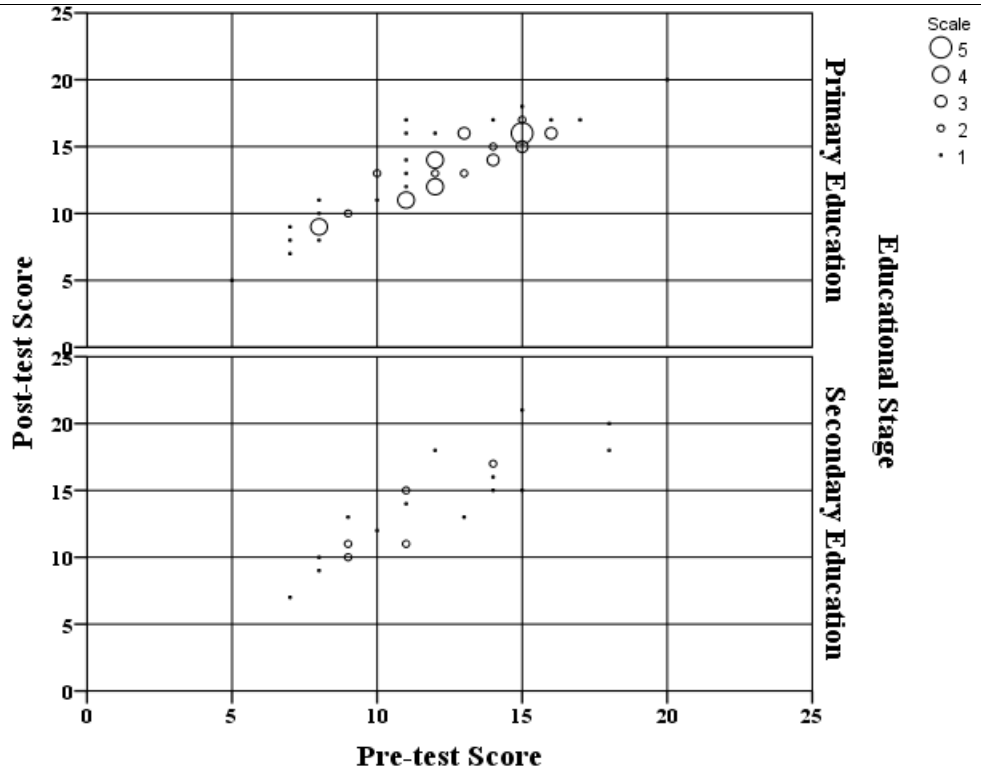

Figure 15. Scatter plot. Pretest Score * Postest Score depending on the Educational Stage

Finally, a Spearman's Rho was calculated to detect correlation between the data-responses measured in an ordinal level, which are the ones corresponding to the following questions of the questionnaire:

- Attractiveness of the website. *What do you think about the website? Do you find it attractive? (1=Not attractive; 2=Just fair; 3=Yes, attractive)*
- Easiness of analysis. *Is it easy to analyze projects with Dr. Scratch? (1=No; 2=Just fair; 3=Yes)*
- Feelings after results. *How did you feel when you saw the results? (1=Bad; 2=Something in between; 3=Good)*
- Readability of results. *Do you understand the information in the results page? (1=No; 2=Not completely; 3=Yes)*
- Willingness of improvement. *After reading the information, do you feel like trying something new? (1=No; 2=Maybe; 3=Yes)*
- Feedback sufficiency. *Using the information that appeared in the help page you selected, try to improve your project by adding something new. Are the ideas and tips in the results pages enough to improve your program? (1=No; 2=Just fair; 3=Yes)*

		Easiness	Feelings	Readability	Willingness	Feedback
Rho of Spearman	**Attractiveness**	**.241***	.063	**.208***	**.195***	.172
	Easiness		**.324****	.005	-.015	-.016
	Feelings			.071	-.086	.120
	Readability				.005	**.294****
	Willingness					**.211***

Table 12. Spearman's Rho correlations for ordinal responses

As shown in Table 12, some significant correlations were found:

- The attractiveness of the web is correlated positively and significantly, albeit with low intensity, with the perception of easiness of analysis, the readability of the results and subsequent motivation to improve.
- The perception of easiness of analysis correlates positively and significantly, although with low to moderate intensity, with good feelings after receiving the results.
- The readability of the results correlates positively and significantly, albeit with low to moderate intensity, with the perception of sufficiency of feedback.
- Finally, the motivation to improve is significantly and positively correlated, although with low intensity, with the perception of sufficiency of feedback.

Conclusions and future work

This paper presents Dr. Scratch, a free/open-source web tool that allows analyzing Scratch projects to automatically assign a CT score as well as to detect potential errors or bad programming habits, aiming to help learners to develop their coding and CT skills as well as to support educators in the evaluation tasks.

In order to assess the effectiveness of Dr. Scratch as a tool to assist programming learners, we run a series of workshops with 109 students between 10 and 14 years from 8 different schools that had prior coding experience with Scratch. The students analyzed one of their Scratch projects with Dr. Scratch, read the information in the feedback report provided by the tool, tried to improve their code following the instructions and finally analyzed again their projects. The results show that, in average, students enhanced their CT Score in 1.45 points, from 12.00/21 to 13.45/21, which represents a statistically significant improvement. In this line, the overall effect size, d = 0.47, indicates a moderate effect that, taking into account that was generated during a one-hour workshop, highlights the real impact that the use of Dr. Scratch had in the coding skills and the development of CT of participants.

The results indicate that the feedback report provided by Dr. Scratch was especially useful for secondary students with an initial medium (*developing*) CT score. However, the tool does not seem to be as helpful for students with an initial high (proficient) score, at least in the tested, one-hour workshop environment. We will devote future work to ascertain how to enhance the feedback provided by the tool. In addition, new research could help us discover if differences of performance between secondary and

primary students are due to the tool itself or are related to the maturation meta-cognitive development of learners.

In the near future we also plan to perform new investigations to try to find correlations with other tools that assess CT of students, such as the Computational Thinking Test (Román-González, 2015), as well as to test the effectiveness of the Dr. Scratch assessment comparing its results with the ones from Primary and Secondary Education expert evaluators. These results would help us adjust and improve the CT analysis operation.

Regarding the features of Dr. Scratch, at the time of writing this paper we are working on several enhancements:

- Plug-ins for browsers: with the plug-ins for Firefox and Chrome, learners will be able to analyze their projects while programming in the Scratch website.
- User accounts: students will be able to keep the log of their analysis to study their evolution on time.
- Teacher accounts: educators will be able to group and follow their students, and to keep track of their progress.
- Translation into new languages: at this moment the tool is available in Spanish and English, but we plan to increase the number of languages with the support of the community.
- Gamification and Social network functionalities: we plan to incorporate new features that allow users to communicate with each other to exchange ideas and challenges.
- 'App' for mobile phones: we are developing an HTML5 'app' to expand the social network and gamification features.

Acknowledgements

The work of all authors has been funded in part by the Region of Madrid under project "eMadrid - Investigación y Desarrollo de tecnologías para el e-learning en la Comunidad de Madrid" (S2013/ICE-2715). The authors are very thankful to the teachers and pupils of the schools and middle schools participating in the investigation: Lope de Vega, Carles Salvador, Miguel de Cervantes, Villa Devoto, Camp de Túria, Mestre Ramón Esteve, Vicente Aleixandre and Fuente San Luis. We would like to thank as well Eva Hu Garres and Mari Luz Aguado for their technical support with Dr. Scratch.

Sent: 31 July 2015
Accepted: 2 September 2015
Published: 15 September 2015

Moreno-León, J., Robles G., & Román-González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED-Revista de Educación a Distancia. Número 46*. 15 de Septiembre de 2015. Consultado el (dd/mm/aaaa) en http://www.um.es/ead/red/46

References

Beyer, S., Rynes, K., Perrault, J., Hay, K., & Haller, S. (2003). Gender differences in computer science students. *SIGCSE Bull.*, *35*(1), 49. doi:10.1145/792548.611930

Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). Hairball: Lint-inspired static analysis of scratch projects. *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE 2013)*, 215-220.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association (AERA 2012)*

Carter, J., & Jenkins, T. (1999). Gender and programming. *SIGCSE Bull.*, *31*(3), 1-4. doi:10.1145/384267.305824

Cohen, J. (1990). Things I have learned (so far). *American Psychologist*, *45*(12), 1304-1312. doi:10.1037/0003-066x.45.12.1304

Ellis, P. (2010). *The essential guide to effect sizes*. Cambridge: Cambridge University Press.

Franklin, D., Conrad, P., Boe, B., Nilsen, K., Hill, C., Len, M., ... Kiefer, B. (2013). Assessment of computer science learning in a scratch-based outreach program. *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE 2013)*, 371-376.

Grover, S., & Pea, R. (2013). Computational thinking in K–12. A review of the state of the field. *Educational Researcher, 42*(1), 38-43.

Kafai, Y. B., Fields, D. A., & Burke, W. Q. (2012). Entering the Clubhouse: Case studies of young programmers joining the online Scratch communities. In A. Dwivedi & S. Clarke (Eds.), *End-User Computing, Development, and Software Engineering: New Challenges* (pp. 279-294). Pennsylvania, PA: IGI Global.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61.

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, *39*(1) 223-227.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education, 23*(3), 239-264.

Moreno, J., & Robles, G. (2014). Automatic detection of bad programming habits in scratch: A preliminary study. *Frontiers in Education Conference (FIE), 2014 IEEE,* 1-4. doi:http://dx.doi.org/10.1109/FIE.2014.7044055

Moreno-León, J., & Robles, G. (2015). Computer programming as an educational tool in the English classroom: A preliminary study. *Proceedings of the 2015 IEEE Global Engineering Education Conference (EDUCON 2015),* 961-966.

Papert, S., & Solomon, C. (1971). *Twenty things to do with a computer*. Retrieved from http://18.7.29.232/bitstream/handle/1721.1/5836/AIM-248.pdf

Resnick, M. (2013). Learn to code, code to learn. *EdSurge, May*.

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

Román-González, M. (2015). Computational Thinking Test: Design Guidelines and Content Validation. *Proceedings of the 7th Annual International Conference on Education and New Learning Technologies (EDULEARN 2015),* 2436-2444.

Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. *Proceedings of the 9th Annual International ACM Conference on International Computing Education Research (ICER 2013),* 59-66.

Wilson, A., Hainey, T., & Connolly, T. (2012). Evaluation of computer games developed by primary school children to gauge understanding of programming concepts. *Proceedings of the 6th European Conference on Games-Based Learning (ECGBL),* 4-5.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Wolz, U., Hallberg, C., & Taylor, B. (2011). Scrape: A tool for visualizing the code of scratch programs. *Poster presented at the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE 2011)*