RED-Revista de Educación a Distancia, 46(10). 15-Sept.-2015
DOI: 10.6018/red/46/10
http://www.um.es/ead/red/46/moreno_robles_es.pdf

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

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

Jesús Moreno-León
Universidad Rey Juan Carlos. España.
jesus.moreno@programamos.es

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

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

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. Con Dr. Scratch, alumnos y educadores pueden comprobar si se han programado correctamente, aprender de sus errores y recibir retroalimentación para mejorar su código, y desarrollar su capacidad de Pensamiento Computacional (PC). Otro 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

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

Introducción

A lo largo de la pasada década hemos visto un renacimiento de la programación y del pensamiento computacional (PC) (Wing, 2006) en las escuelas (Lye & Koh, 2014). El uso educativo de la programación, que fue introducido en los años 70 y 80 del siglo pasado - principalmente a través del lenguaje de programación Logo (Papert & Solomon, 1971)-, ha vuelto con fuerza gracias a nuevos lenguajes visuales como Alice, Kodu y, especialmente, Scratch, que permiten a estudiantes jóvenes programar aplicaciones sin la necesidad de aprender la compleja sintaxis de los lenguajes de programación tradicionales.

Scratch (Resnick et al., 2009) es un lenguaje de programación diseñado para niños a partir de 6 años, que también ofrece un sitio web donde los usuarios pueden compartir sus proyectos e intercambiar ideas o sugerencias con otros (jóvenes) programadores. Scratch se está usando masivamente en todo el mundo, con más de siete millones de proyectos compartidos en su repositorio[1]. Uno de los objetivos principales de Scratch es que la programación se utilice como una herramienta para desarrollar otras habilidades y para mejorar el aprendizaje de otras disciplinas (Resnick, 2013). Como resultado, Scratch se está usando en actividades extraescolares (Kafai, Fields, & Burke, 2012) y en todos los niveles de la educación formal, en colegios (Moreno-León & Robles, 2015), institutos (Meerbaum-Salant, Armoni, & Ben-Ari, 2013) e incluso universidades (Malan & Leitner, 2007) de todo el mundo.

Sin embargo, existe un déficit de herramientas que ayuden a los docentes a evaluar los programas del alumnado y a medir el desarrollo de su PC. Esta situación se debe parcialmente a que no existe un consenso en la definición de PC ni al modo en que éste debe introducirse en el currículo (Grover & Pea, 2013). En este artículo se presenta Dr. Scratch, una herramienta web libre/de código abierto que analiza proyectos Scratch para (1) ofrecer retroalimentación a docentes y estudiantes, y (2) asignar una puntación de PC a los proyectos. Los aprendices pueden usar esta información para mejorar sus programas y ser conscientes de cómo desarrollar sus habilidades de programación. Para probar la efectividad de Dr. Scratch hemos desarrollado una serie de talleres para medir el impacto de su uso en el aprendizaje. Los resultados obtenidos muestran un impacto positivo y nos dan ideas de cómo proceder para seguir desarrollando la herramienta en el futuro.

La estructura del artículo es la siguiente: la sección de Antecedentes repasa distintas propuestas y herramientas que tratan de apoyar a educadores en el proceso de evaluación del PC del alumnado; a continuación se explican las funcionalidades incluidas en Dr. Scratch; el enfoque seguido para preparar los talleres en los que se ha probado Dr. Scratch con estudiantes se detalla en la sección de Metodología; los resultados de los talleres, tanto cualitativos como cuantitativos, se muestran en la sección de Resultados; por último, en las Conclusiones se realiza un resumen del estudio, se discuten las limitaciones de la herramienta y se presentan algunas de las nuevas funcionalidades en las que está trabajando el equipo de desarrollo.

Antecedentes

Existe un déficit de herramientas que ayuden a los docentes en la evaluación del desarrollo del PC y en la corrección de los proyectos programados por sus estudiantes. En relación al lenguaje de programación Scratch, varios autores han propuesto distintos enfoques para evaluar el desarrollo del PC analizando los proyectos de los aprendices, pero la mayoría de estas aproximaciones se basan exclusivamente en un análisis manual.

Wilson, Hainey, y Connolly (2012) proponen un esquema para establecer el nivel de competencia de programación demostrado por un estudiante mediante el análisis de un proyecto en términos de conceptos de programación (como hilos, instrucciones condicionales o variables), organización de código (nombres de variables, nombres de objetos y bloques extraños), y diseño y usabilidad (como funcionalidad u originalidad, entre otros).

En esta línea, Seiter y Foreman (2013) desarrollaron el Modelo de Progresión de Pensamiento Computacional

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

Temprano, un marco para medir el PC de estudiantes de primaria trabajando con Scratch, para lo que sintetizaban "evidencias medibles del trabajo del estudiante con patrones de diseño, más abstractos y amplios, que son luego mapeados en conceptos de PC" (Seiter & Foreman, 2013, p. 59).

En el artículo *"New frameworks for studying and assessing the development of computational thinking"*, Brennan y Resnick (2012) introdujeron una estrategia basada en el análisis del *portfolio* de proyectos utilizando una herramienta de visualización llamada Scrape (Wolz, Hallberg, & Taylor, 2011), que parece no estar disponible en la actualidad, aunque su propuesta se completa con entrevistas sobre los proyectos y diseño de escenarios.

Con el objetivo de ayudar a los docentes con una herramienta que pudiera ser utilizada para automatizar parcialmente la evaluación de proyectos Scratch, Boe et al. (2013) desarrollaron Hairball, una analizador de código estático que detecta problemas potenciales en los programas, como código que nunca se ejecuta, mensajes que no recibe ningún objeto, o atributos que no se inicializan correctamente. Tras dos semanas de un campamento de verano con Scratch, se usó Hairball para medir los conocimientos de informática sobre programación dirigida por eventos, inicialización de estado y paso de mensajes (Franklin et al., 2013).

La arquitectura de Hairball, que está basada en plug-ins, es ideal para añadir nuevas funcionalidades. En un trabajo previo, los autores desarrollamos un par de plug-ins para detectar dos malos hábitos de programación que habíamos percibido de forma frecuente en nuestro trabajo como instructores con estudiantes de secundaria (Moreno & Robles, 2014):

- *convention.SpriteNaming*[2] analiza un proyecto Scratch para comprobar si los nombres de los objetos comienzan con el texto *Sprite* (*Objeto)*, lo que indica que el programador no ha modificado el nombre que Scratch asigna por defecto a los nuevos objetos. Hay que mencionar que aunque el uso de nombres por defecto no produce ningún error si el programa se implementa correctamente, sí dificulta su legibilidad, especialmente cuando el número de objetos es grande (por ejemplo, mayor que diez).
- *duplicate.DuplicateScripts*[3] analiza un proyecto Scratch para encontrar programas duplicados, que se repiten dentro de los programas de un objeto. Para este tipo de estructuras se debería utilizar la opción de crear bloques personalizados.

Tratando de comprobar si estos malos hábitos de programación son también habituales en los proyectos compartidos en el repositorio de Scratch, descargamos aleatoriamente 100 proyectos, detectando que el 79% de los proyectos inspeccionados incluía nombres no personalizados, mientras que el 62% incluía código repetido (Moreno & Robles, 2014). Estos datos nos animaron a desarrollar una herramienta que ayudara a educadores y aprendices a detectar problemas en su código para mejorar sus habilidades de programación.

El hecho de que Hairball se ejecuta desde la línea de comandos, ya que se basa en scripts programados en *Python* que el evaluador tiene que lanzar manualmente, hace que no sea una herramienta adecuada para docentes que no tienen experiencia en este tipo de entornos, y menos aún para estudiantes jóvenes. Por este motivo nos decidimos a crear un servicio web, Dr. Scratch, que permita realizar análisis de proyectos de forma sencilla.

Presentando Dr. Scratch

Dr. Scratch[4] es una aplicación web libre/de código abierto que permite analizar de forma simple proyectos Scratch utilizando plug-ins de Hairball, así como obtener retroalimentación que puede usarse para mejorar las habilidades de programación y desarrollar el PC. Para analizar un proyecto con Dr. Scratch puede subirse un archivo .sb o .sb2, ya que la herramienta soporta tanto proyectos de la versión Scratch 1.4 como 2.0, o bien se puede copiar directamente la URL del proyecto. La posibilidad de analizar proyectos desde su URL ha sido

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

implementada utilizando *getsb2*[5].

Cuando se analiza un proyecto Scratch, Dr. Scratch informa al usuario sobre el grado de desarrollo de PC demostrado en ese proyecto, asignando una puntuación de PC. Al estar basado en Hairball, Dr. Scratch detecta ciertos malos hábitos de programación o errores potenciales, como el uso de nombres no significativos, la repetición de código, código que no llega a ejecutarse y la inicialización incorrecta de atributos de objetos.

Para asignar la puntuación de PC, Dr. Scratch deduce la competencia demostrada por el programador en los siguientes siete conceptos: abstracción y descomposición de problemas, pensamiento lógico, sincronización, paralelismo, nociones algorítmicas de control de flujo, interactividad con el usuario y representación de la información. La evaluación de la competencia en cada uno de estos conceptos sigue las reglas de la Tabla 1, que fue diseñada en base a las propuestas presentadas en la sección de Antecedentes, reutilizando algunas de sus ideas con la ayuda de docentes de distintos niveles educativos que utilizan Scratch en sus clases.

En función de la puntuación de PC, que va de 0 a 21 puntos, la información mostrada en las pantallas de resultados es distinta. Así, si el nivel de PC es bajo, se asume que el usuario es un programador novato y, en consecuencia, la herramienta solamente muestra información básica sobre las mejoras más importantes que pueden realizarse en el código. Según va aumentando la puntuación, Dr. Scratch muestra más información de los proyectos analizados. Por tanto, usuarios avanzados reciben un informe de retroalimentación con toda la información disponible, tanto en términos de habilidades de PC como en lo relativo a los malos hábitos de programación. Las Figuras 1 y 2 ilustran las diferencias en cantidad y complejidad de la información mostrada en pantalla en función de la puntuación de PC.

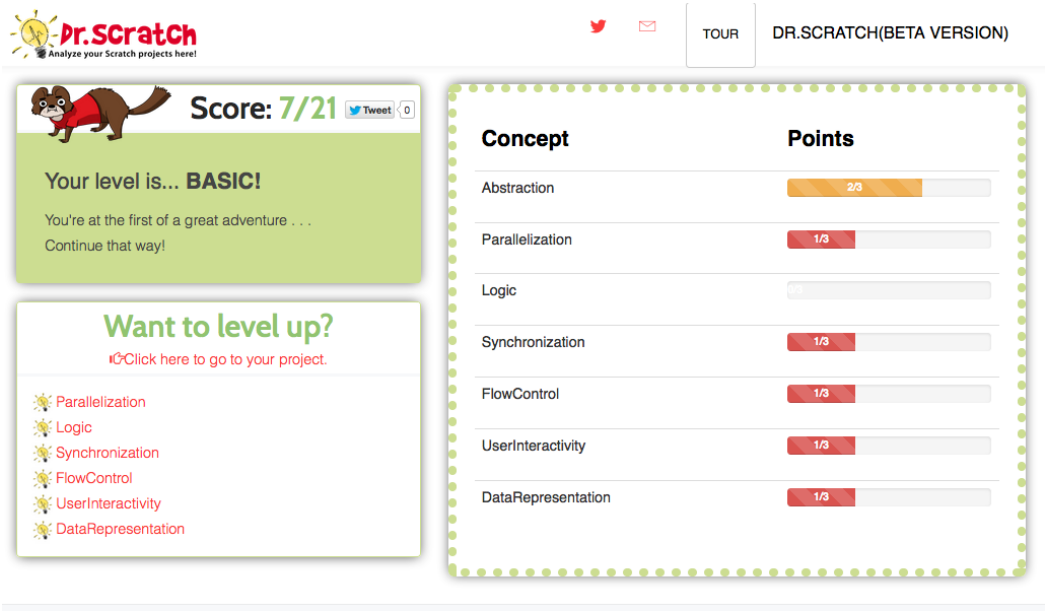

Fig. 1. Resultados de un análisis con Dr. Scratch de un proyecto con un nivel bajo de PC

La Figura 3 puede usarse para ilustrar el funcionamiento de la evaluación de PC. Así, siguiendo las reglas de la Tabla 1, el primer programa de la imagen sería catalogado como *básico* en lo relativo a la representación de la información, ya que modifica algunas de las propiedades del objeto (posición y orientación). El segundo programa, sin embargo, se consideraría que tienen un nivel *en desarrollo*, ya que utiliza variables. Por último, el tercer programa mostraría un nivel *competente* en este concepto, ya que realiza una operación con una lista.

En aquellos aspectos donde existe posibilidad de mejora, la herramienta ofrece enlaces a información que

[5] https://github.com/nathan/getsb2

puede utilizarse para desarrollar la competencia en cuestión. Por ejemplo, si un proyecto ha obtenido un punto en paralelismo, Dr. Scratch ofrece un enlace a código de ejemplo junto con una explicación sobre cómo conseguir que varias acciones ocurran al mismo tiempo en un programa (ver Figura 4).

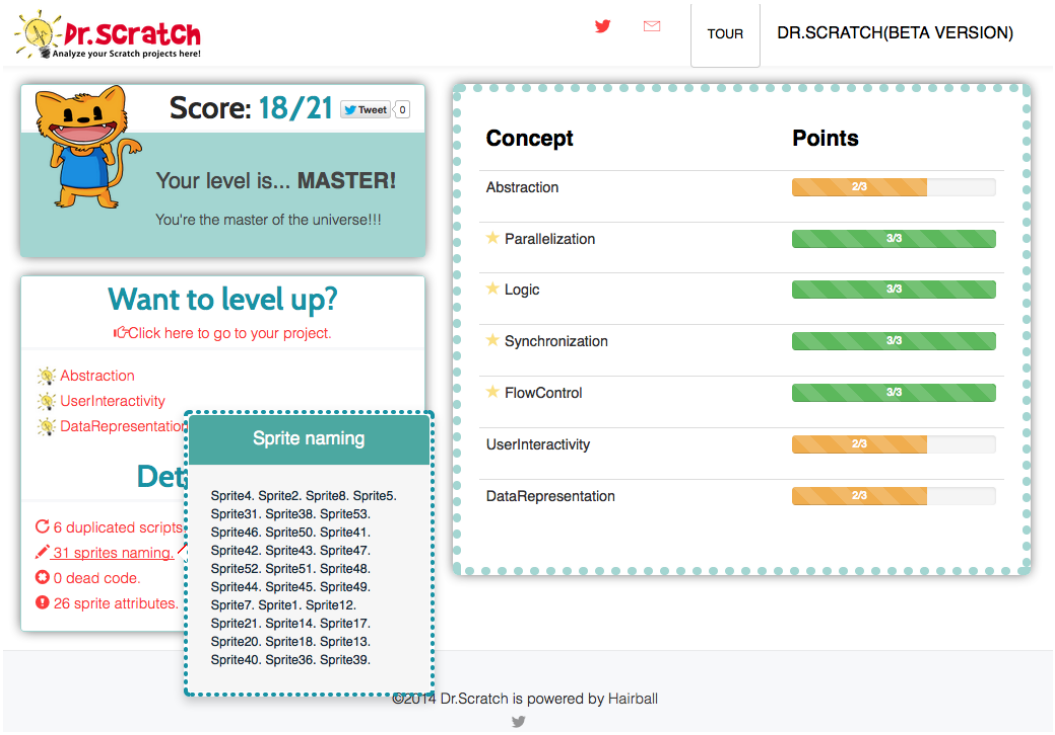

Fig. 2. Resultados de un análisis con Dr. Scratch de un proyecto con un nivel alto de PC

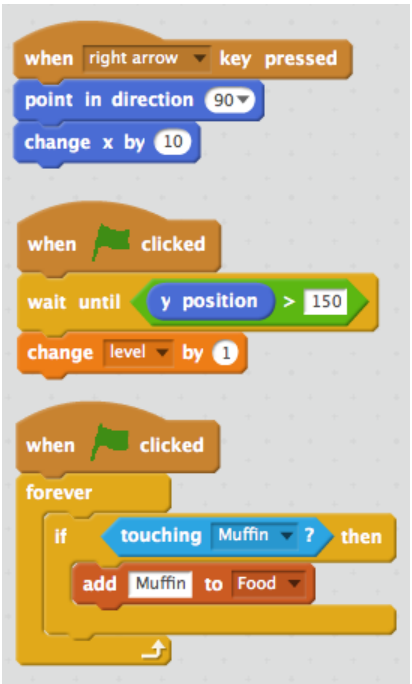

Fig. 3. Distintos niveles de competencia para representación de la información: *básico* (arriba), *en desarrollo* (en medio) y *competente* (abajo).

Aspecto de PC	Nivel de competencia			
	Ninguno (0)	**Básico (1 punto)**	**En desarrollo (2 puntos)**	**Competente (3 puntos)**
Abstracción y descomposición de problemas	-	Más de un programa y más de un objeto	Definición de bloques propios	Uso de clones
Paralelismo	-	Dos programas en 'bandera verde'	Dos programas en 'tecla presionada', dos programas en 'al presionar' el mismo objeto	Dos programas en 'cuando reciba mensaje', crear clon, dos programas en 'cuando %s es > %', dos programas en 'cuando el escenario cambie a'
Pensamiento lógico	-	Si	Si - sino	Operaciones lógicas
Sincronización	-	Esperar	Enviar, cuando reciba mensaje, parar todos, parar programas, parar programas del objeto	Esperar hasta, cuando el escenario cambie a, enviar y esperar
Control de flujo	-	Secuencia de bloques	Repetir, por siempre	Repetir hasta
Interactividad con el usuario	-	Bandera verde	Tecla presionada, objeto presionado, preguntar y esperar, bloques de operaciones con ratón	Cuando %s es >%s, vídeo, audio
Representación de la información	-	Modificadores de propiedades de objetos	Operaciones con variables	Operaciones con listas

Tabla 1. Nivel de competencia para cada aspecto de Pensamiento Computacional

Metodología

Con el objetivo de medir la efectividad de Dr. Scratch como herramienta de ayuda a jóvenes aprendices de programación, se organizaron una serie de talleres con estudiantes entre 10 y 14 años de 8 centros educativos españoles de primaria y secundaria (ver Figura 5). Estos estudiantes ya habían aprendido previamente a programar con Scratch durante varias semanas en sus centros.

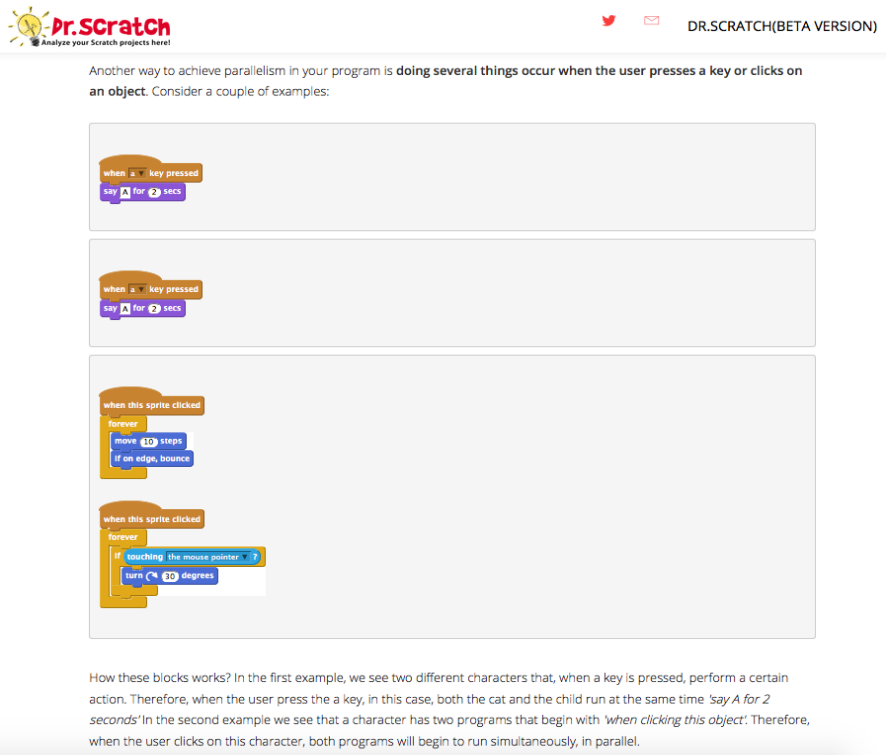

Fig. 4. Ideas y trucos ofrecidos por Dr. Scratch para mejorar proyectos incorporando paralelismo

Durante los talleres, de una hora de duración, se entregó un cuestionario a los estudiantes con algunas preguntas que tenían que responder mientras realizaban una serie de tareas. Las tareas y preguntas se detallan a continuación:

1) Visita la web de Dr. Scratch:
   a) ¿Qué te parece la web? ¿Te resulta atractiva?
   b) Tras leer la información del sitio web, ¿para qué crees que sirve Dr. Scratch?
2) Analiza uno de tus proyectos Scratch con Dr. Scratch
   a) ¿Te ha resultado sencillo analizar tu proyecto con Dr. Scratch?
   b) ¿Cuál ha sido tu puntuación?
   c) Según Dr. Scratch, ¿a qué nivel corresponde esa puntuación?
   d) ¿Cómo te has sentido al comprobar los resultados?
   e) ¿Por qué?
3) Desde la página de resultados, tras analizar un proyecto, haz clic sobre alguno de los enlaces para recibir información que pueda ayudarte a mejorar tu proyecto.
   a) Escribe el título de la página en la que has hecho clic.
   b) ¿Entiendes la información de la página de resultados?
   c) Tras leer la información, ¿tienes ganas de probar algo nuevo?
4) Utilizando la información de la página de resultados que has seleccionado, trata de mejorar tu proyecto añadiendo algo nuevo.
   a) ¿Son suficientes las ideas y trucos de la página de resultados para mejorar tu programa?
   b) Tras realizar algunas modificaciones, analiza de nuevo tu proyecto con Dr. Scratch. ¿Cuál es la nueva puntuación?
5) ¿Tienes algún otro comentario?

Fig. 5. Taller de Dr. Scratch en el colegio Lope de Vega, Madrid.

Características de la muestra del estudio

La Tabla 2 muestra las características de la muestra del estudio en relación a la edad de los participantes, formado por un grupo de 109 estudiantes entre 10 y 14 años. La edad media de la muestra fue 11,50, mientras que la mediana fue 11 y la moda 10.

N	**Válido**	109
	Perdidos	0
Media		11,50
Mediana		11,00
Moda		10
Desviación estándar		1,392
Varianza		1,937
Mínimo		10
Máximo		14

Tabla 2. Edad de los estudiantes participantes en la investigación

La Figura 6 muestra el porcentaje de estudiantes por cada grupo de edad. Como puede verse, una mayoría de participantes tenía 10 u 11 años.

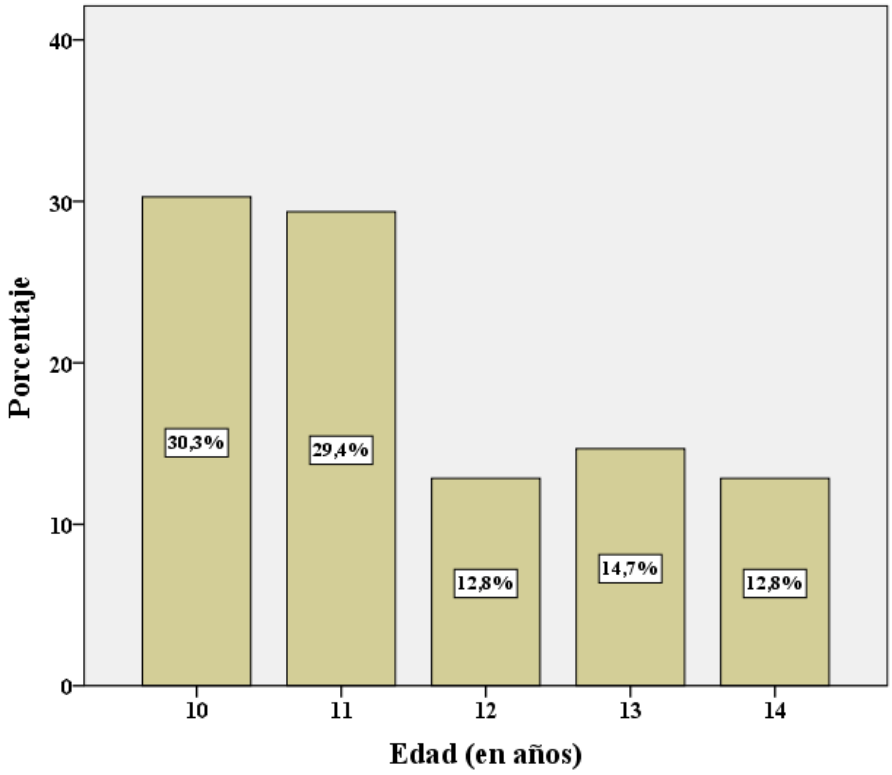

Figura 6. Porcentaje de estudiantes por grupo de edad (en años)

En relación al sexo, la Tabla 3 muestra el porcentaje de chicos y chicas participantes en la investigación; el 57,8% fueron chicos, mientras que el 42,2% fueron chicas. Esta diferencia se explica por el hecho de que en algunos de los institutos el experimento se desarrolló en asignaturas optativas como informática o tecnología, donde es habitual que exista una mayoría de chicos. Por otro lado, el género de los participantes no se recogió en uno de los centros educativos, lo que explica los 19 registros perdidos de la Tabla 3.

Aunque existe literatura abundante que investiga acerca de temas de género en la educación informática (Beyer, Rynes, Perrault, Hay & Haller, 2003) y la programación (Carter & Jenkins, 1999), este tema esta fuera del alcance de nuestra investigación. No obstante, tenemos planificados futuros estudios para comprobar si chicos y chicas reaccionan y/o aprenden de forma distinta con Dr. Scratch.

Finalmente, en relación a la etapa educativa de los alumnos, tal como puede comprobarse en la Tabla 4, una mayoría de los participantes se encontraba matriculado en educación primaria, aunque también tuvimos un grupo significativo de alumnos provenientes de educación secundaria.

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	**Chico**	52	47,7	57,8	57,8
	Chica	38	34,9	42,2	100,0
	Total	90	82,6	100,0	
Perdidos	**Sistema**	19	17,4		
Total		109	100,0		

Tabla 3. Porcentaje de estudiantes por sexo

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	**Educación Primaria**	75	68,8	68,8	68,8
	Educación Secundaria	34	31,2	31,2	100,0
	Total	109	100,0	100,0	

Tabla 4. Porcentaje de estudiantes por etapa educativa

Resultados

Las Figuras 7, 8, 9, 10 y 11 muestran las respuestas de los alumnos a algunas de las preguntas del cuestionario. De acuerdo con las respuestas, una mayoría de alumnos consideran que el sitio web de Dr. Scratch es atractivo, como puede verse en la Figura 7, y la gran mayoría de los estudiantes creen que analizar proyectos con Dr. Scratch es sencillo (Figura 8). En relación a sus sentimientos tras analizar los proyectos, cuyas respuestas se muestran en la Figura 9, una mayoría de aprendices se sintieron bien cuando vieron la puntuación de PC, aunque un 3% de los que respondieron indicaron que se sintieron mal. En lo relativo a la información mostrada por Dr. Scratch, la Figura 10 indica que la mayoría de los alumnos fueron capaces de entenderla; sin embargo, un 5,6% de los aprendices indicaron que no comprendieron la información recibida. Por último, teniendo en cuenta que uno de los objetivos de Dr. Scratch es estimular el auto-aprendizaje a través de un entorno *gamificado*, estábamos interesados en la reacción de los estudiantes al recibir el informe de retroalimentación. En este sentido, la Figura 11 muestra que Dr. Scratch promueve el deseo de querer mejorar las habilidades de programación.

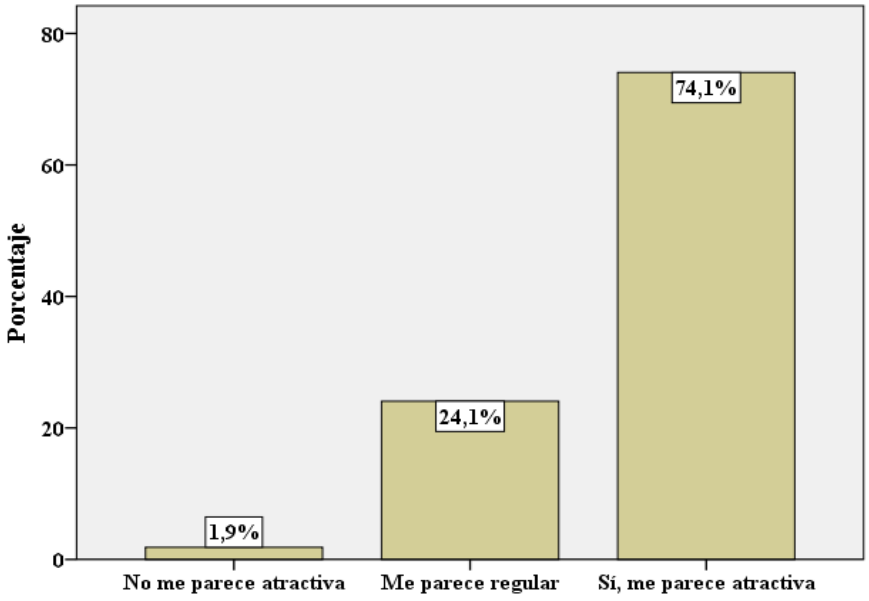

Figura 7. ¿Qué te parece la página? ¿Te parece atractiva?

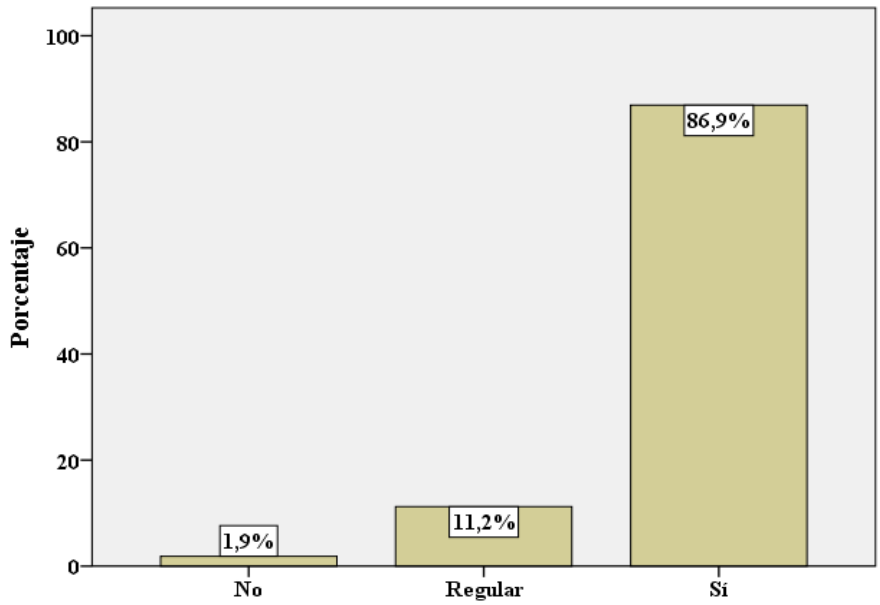

Figura 8. Analiza uno de tus proyectos Scratch con Dr. Scratch. ¿Te ha resultado sencillo realizar el análisis?

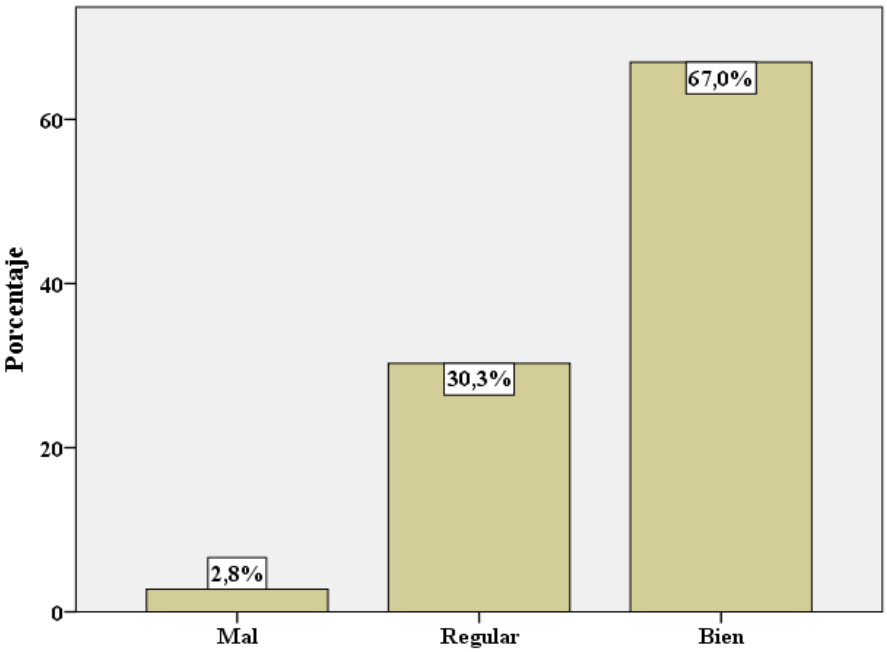

Figura 9. ¿Cómo te has sentido al ver los resultados?

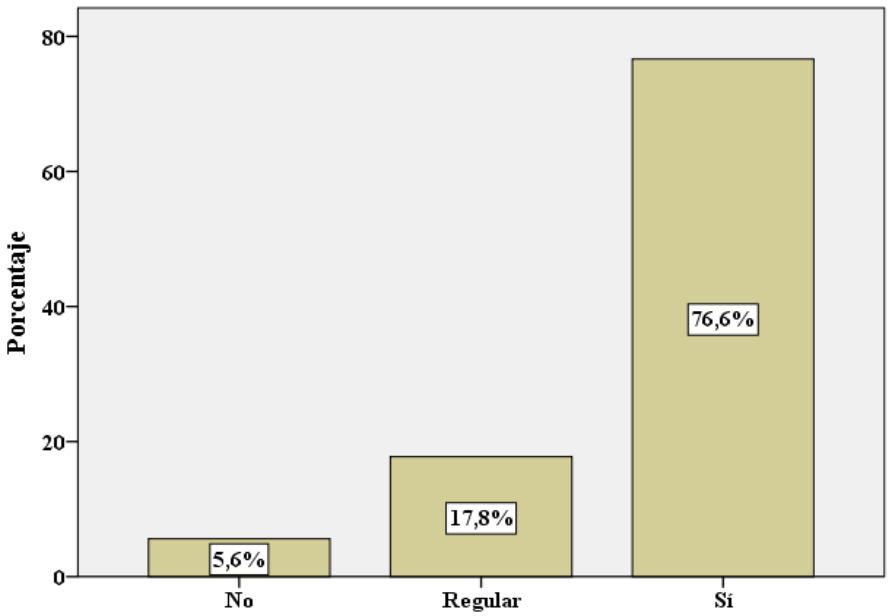

Figura 10. ¿Te resulta comprensible la información que se muestra en esta nueva página (página de resultados)?

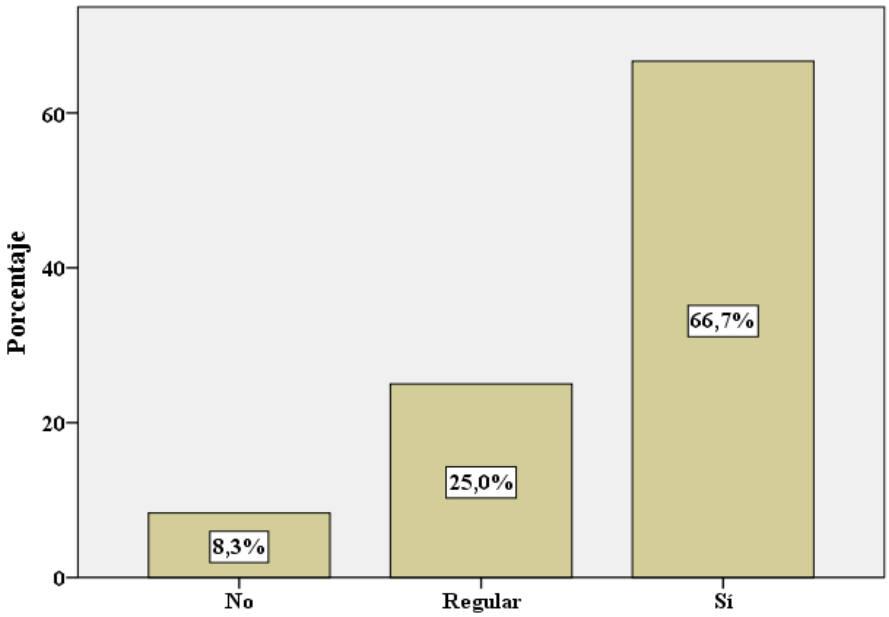

Figura 11. Tras leer la información, ¿tienes ganas de mejorar tu proyecto probando algo nuevo en Scratch?

La Tabla 5, y las Figuras 12 y 13 muestran los resultados del análisis de los proyectos Scratch desarrollados por los estudiantes antes y después de recibir los informes de retroalimentación proporcionados por Dr.

Scratch. Como puede comprobarse, hubo un incremento en los resultados, ya que la media de los análisis pre-test es 11,82, mientras que la media de los análisis posteriores es 13,52.

Para ser capaces de probar que la mejora experimentada por los estudiantes fue estadísticamente significativa, realizamos una prueba *t* para muestras emparejadas, estableciendo un nivel de confianza del 95% ($\alpha = 0,05$) para nuestras decisiones estadísticas.

La Tabla 6 muestra las estadísticas de los 88 alumnos que indicaron correctamente la puntuación del análisis pre y post de sus proyectos. Hubo 21 registros no completados de los 109 alumnos participantes, debido a que las puntuaciones pre-test o post-test no fueron indicadas correctamente. Son varias las circunstancias que concurren en este sentido para explicar las 21 pérdidas: desde estudiantes que indicaron su nivel de PC (bajo, en desarrollo, competente) en lugar de su puntuación, hasta fallos de conexión a internet en mitad de la prueba o incluso errores de la propia web de Dr. Scratch. De los 88 registros completos, el valor de la puntuación de PC medio se incrementó de 12,00 en el pre-test a 13,45 en el post-test.

		Antes del feedback de Dr. Scratch (Puntuación Pre-test)	***Después del feedback de Dr. Scratch (Puntuación Post-test)***
Media		11,82	13,52
Mediana		12,00	14,00
Moda		[11, 12, 15]	16
Desviación		3,093	3,257
Asimetría		,028	-,171
Mínimo		5	5
Máximo		20	21
Percentiles	**10**	8,00	9,00
	20	9,00	10,40
	30	10,00	11,00
	40	11,00	13,00
	50	12,00	14,00
	60	13,00	15,00
	70	14,00	16,00
	80	15,00	16,00
	90	15,60	17,00

Tabla 5. Resultados de los análisis de los proyectos Scratch, antes y después de leer el feedback de Dr. Scratch

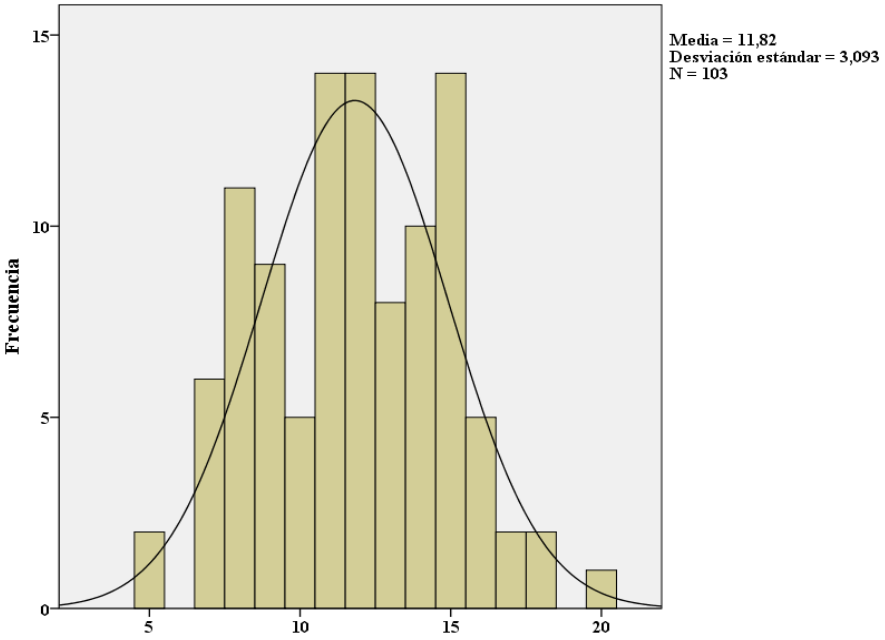

Figura 12. ¿Qué puntuación has obtenido? (Puntuaciones Pre-test)

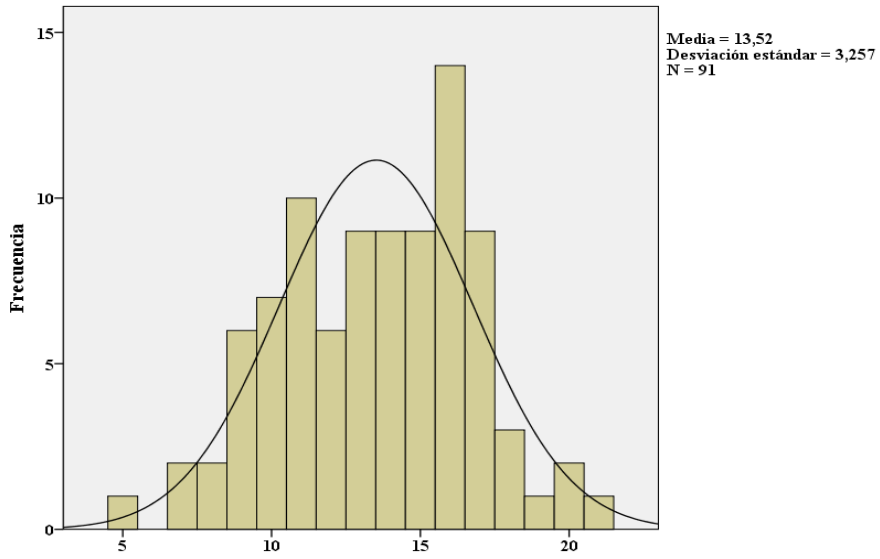

Figura 13. Tras realizar las mejoras siguiendo las instrucciones de la página de ayuda, analiza de nuevo tu proyecto con Dr. Scratch. ¿Qué puntuación has obtenido ahora? (Puntuaciones Post-test)

Estadísticas de muestras emparejadas				
	Media	**N**	**Desviación estándar**	**Media de error estándar**

Par 1	**Puntuación Pre-test**	12,00	88	2,983	,318
	Puntuación Post-test	13,45	88	3,216	,343

Tabla 6. Estadísticas de muestras emparejadas

Prueba de muestras emparejadas									
		Diferencias emparejadas					**t**	**gl**	**Sig. (bilateral)**
		Media	**Desviación estándar**	**Media de error estándar**	**95% de intervalo de confianza de la diferencia**				
					Inferior	**Superior**			
Par 1	**Puntuación Pretest – Puntuación Postest**	-1,455	1,523	,162	-1,777	-1,132	-8,959	87	**,000**

Tabla 7. Prueba *t* para muestras emparejadas en pre-test y post-test

Los resultados de la prueba *t* para muestras emparejadas se muestran en la Tabla 7. Dado que p(t) = 0,000 << 0,05, la hipótesis nula de igualdad de medias es rechazada y, por consiguiente, podemos afirmar que hay diferencias significativas entre las pruebas pre y post, lo que indica que el uso de Dr. Scratch ayudó a los alumnos participantes a desarrollar su PC.

Para tratar de establecer el impacto real de usar Dr. Scratch en el desarrollo del PC, se considera que el tamaño del efecto es un buen indicador, ya que es independiente del tamaño de la muestra, y puede ser calculado de acuerdo a la siguiente fórmula (Cohen, 1990):

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

El tamaño del efecto para nuestra investigación fue de 0,47, lo que se considera un efecto moderado (Ellis, 2010). No obstante, hay que señalar que este efecto se produjo tras solo una hora de tratamiento, que consistió en el taller, lo que subraya el impacto que Dr. Scratch tuvo en los aprendices, y llama la atención sobre su potencial como herramienta para fomentar el PC al ser utilizada como herramienta de apoyo en un curso de programación.

Como era esperado, existe una correlación positiva y muy significativa entre las puntuaciones del pre-test y el post-test, tal como se muestra en la Tabla 8.

Correlaciones de muestras emparejadas				
		N	Correlación	Sig.
Par 1	Puntuación Pretest * Puntuación Postest	88	,882	,000

Tabla 8. Correlación de puntuaciones emparejadas en pre-test y post-test

La Figura 14, por su parte, presenta el diagrama de dispersión para la muestra al comparar las puntuaciones pre-test y post-test de cada sujeto. Como puede comprobarse, todos los casos caen sobre el área de mejora, lo que significa que las puntuaciones en el post-test fueron iguales o mejores que las del pre-test para los 88 estudiantes; por tanto, ninguno de los aprendices redujo su puntuación durante el experimento.

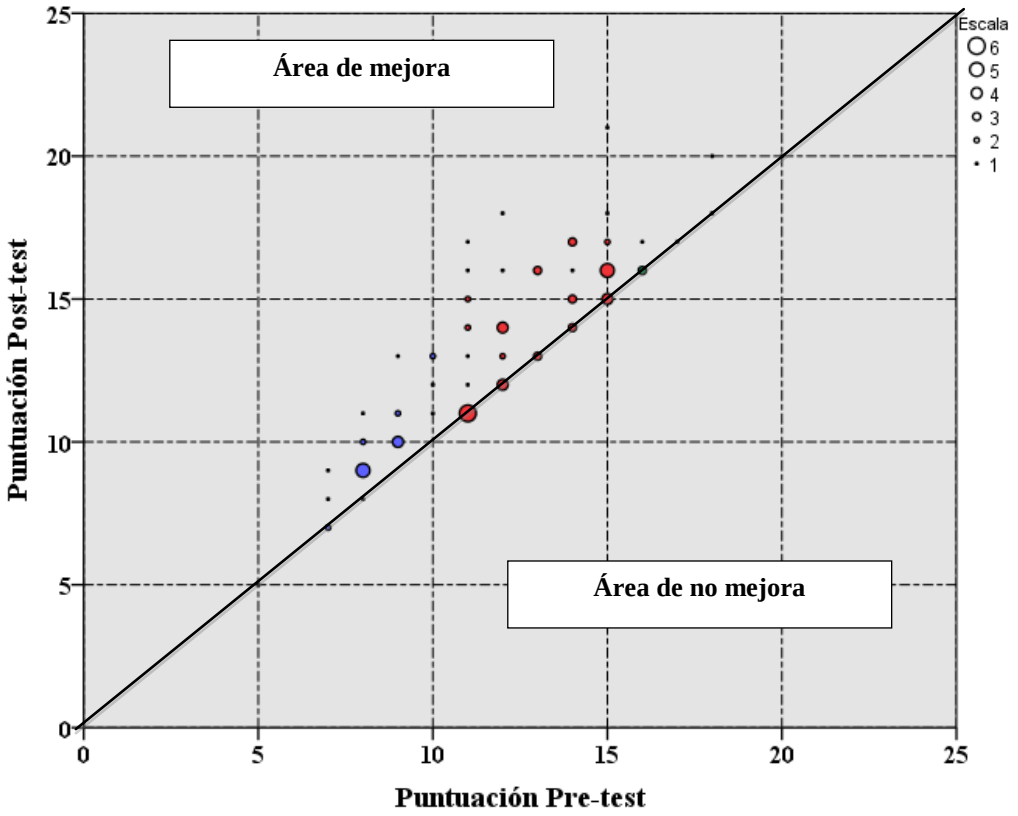

Figura 14. Diagrama de dispersión con las puntuaciones pre-test y post-test para cada sujeto. Los sujetos cuya puntuación post-test es mayor que la puntuación pre-test quedan representados en el 'área de mejora'; los sujetos cuya puntuación post-test es menor que la puntuación pre-test quedan representado en el 'área de no mejora'

Se han realizado varios análisis de significatividad de las diferencias de pretest-postest en función de varios factores. En primer lugar, el análisis de significatividad se realizó en base a la puntuación inicial. Así, dividimos la muestra en tres subconjuntos en función de la puntuación del pre-test:

- Sub-muestra "Puntuación inicial baja": puntuación pre-test ≤ 10 (color azul en la Figura 14)
- Sub-muestra "Puntuación inicial media": puntuación pre-test ≤ 15 (color rojo en la Figura 14)
- Sub-muestra "Puntuación inicial alta": puntuación pre-test ≤ 21 (color verde en la Figura 14)

La Tabla 9 muestra la significatividad y los valores de correlación tanto para la muestra total como para cada una de las sub-muestras. Los resultados indican que el uso de Dr. Scratch generó mejoras significativas en el desarrollo del PC para la sub-muestra "puntuación inicial media" (11-15) y para la sub-muestra "puntuación inicial baja" (0-10), aunque la mejora fue ligeramente inferior en este segundo caso. Sin embargo, Dr. Scratch no generó una mejora significativa para la sub-muestra "puntuación inicial alta" (16-21). Estos resultados están, en cierto modo, en línea con lo que era esperado, ya que indican que los estudiantes con niveles bajo y medio, donde existe más espacio de mejora, son capaces de hacer uso de la información ofrecida por Dr. Scratch para aumentar su puntuación en tan solo una hora, mientras que la herramienta parece menos útil para aquellos alumnos con una puntuación inicial alta, donde los incrementos son más difíciles de conseguir. Las diferencias entre los niveles inicial y medio también entran dentro de lo esperado, hasta cierto punto, ya que los aprendices novatos tienen más dificultades para solventar los problemas que surgen al dar los primeros pasos con Scratch y para comprender la información que ofrece la herramienta. No obstante, Dr. Scratch ofrece una retroalimentación que depende de la puntuación de PC, ya que en las primeras pruebas que se realizaron con aprendices los autores comprobaron que ofrecer mucha información en las primeras fases era contraproducente. Investigaciones futuras nos ayudarán a seguir modificando y adaptando los informes de resultados en esta línea.

	N	**Diferencia post - pre**	**t**	**p (t)**	**¿Diferencia significativa al 95%?**	**r**	**p (r)**
Muestra total	88	1,455	8,959	0,000 < 0,05	Sí	0,88	0,000
Sub-muestra NI Bajo	25	1,440	6,896	0,000 < 0,05	Sí	0,89	0,000
Sub-muestra NI Medio	55	1,618	6,951	0,000 < 0,05	Sí	0,61	0,000
Sub-muestra NI Alto	8	0,375	1,426	0,197 > 0,05	No	0,90	0,002

Tabla 9. Valores de significatividad y correlación en función de la puntuación pre-test

Con el objetivo de comprobar si el uso de Dr. Scratch tuvo un efecto distinto en alumnos de primaria y secundaria, se realizó un análisis de significatividad sobre las diferencias pretest-posttest en función de la etapa educativa. La Tabla 10 muestra la significatividad, el tamaño del efecto y los valores de correlación, tanto para la muestra total como para cada sub-muestra (educación primaria y secundaria). Como puede verse, tanto en primaria como en secundaria el uso de Dr. Scratch generó una mejora significativa, aunque el incremento fue mayor en el segundo caso.

	N	Diferencia post - pre	t	p (t)	¿Diferencia significativa al 95%?	r	p (r)	d	Tamaño del efecto
Muestra total	88	1,455	8,959	0,000 < 0,05	Sí	0,88	0,000	0,47	Moderado
Sub-muestra Primaria	64	1,234	7,294	0,000 < 0,05	Sí	0,90	0,000	0,40	Moderado (-)
Sub-muestra Secundaria	24	2,042	5,540	0,000 < 0,05	Sí	0,87	0,000	0,60	Moderado (+)

Tabla 10. Valores de significatividad y correlación en función de la etapa educativa

El análisis de covarianza, que se muestra en la Tabla 11, confirma la significatividad del efecto de la etapa educativa en el post-test controlando las diferencias base en el pre-test, ya que p ($F_{\text{Etapa Educativa}}$) = 0,03 < 0,05

Pruebas de efectos inter-sujetos					
Variable dependiente: Puntuación Post-test					
Origen	**Tipo III de suma de cuadrados**	**gl**	**Cuadrático promedio**	**F**	**Sig.**
Modelo corregido	710,667[a]	2	355,333	159,678	,000
Interceptación	22,806	1	22,806	10,248	,002
Puntuación Pre-test	708,541	1	708,541	318,401	,000
Etapa Educativa	10,801	1	10,801	4,854	**,030**
Error	189,151	85	2,225		
Total	16830,000	88			
Total corregido	899,818	87			
a. R al cuadrado = ,790 (R al cuadrado ajustada = ,785)					

Tabla 11. Análisis de Covarianza (ANCOVA). Variable dependiente: Puntuación Post-Test

La Figura 15 puede utilizarse para ilustrar la mayor mejora de los alumnos de secundaria. Como se puede comprobar, existe una mayor densidad de casos con la misma puntuación pretest-posttest (alumnos que no mejoran, aunque tampoco empeoran, tras usar Dr. Scratch) en la educación primaria, mientras que en secundaria la mayor densidad se traslada al área de mejora. Estos resultados nos animan a planificar investigaciones futuras para determinar si las diferencias entre estudiantes de primaria y secundaria se deben a nuestra herramienta, debido al lenguaje utilizado y a los ejemplos incluidos en el informe de resultados, o más bien se encuentran relacionadas con el desarrollo meta-cognitivo que, en consecuencia, mejora con la edad.

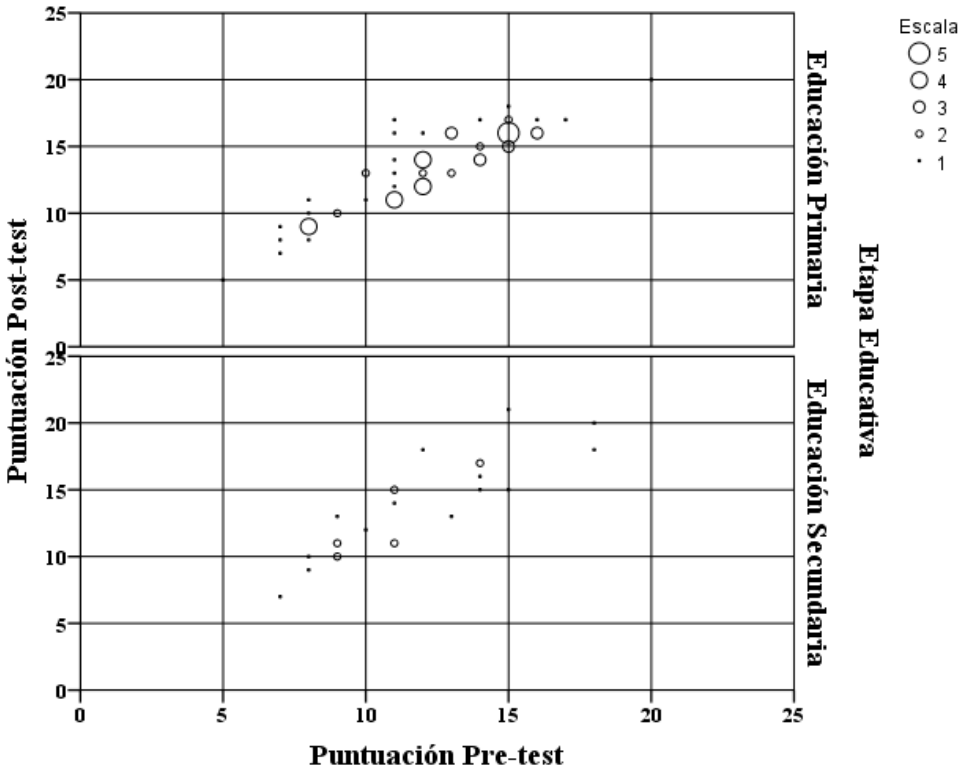

Figura 15. Diagrama de dispersión Puntuación Pre-test * Puntuación Post-test según etapa educativa

Por último, se calculó el coeficiente de correlación *Rho* de *Spearman* para detectar correlaciones entre las respuestas del cuestionario medidas en un nivel ordinal, que son las que se corresponden con las siguientes preguntas:

- Atractivo del sitio web. *¿Qué te parece la web? ¿Te resulta atractiva? (1=No me parece atractiva; 2=Regular; 3=Sí, me parece atractiva)*
- Sencillez del análisis. *¿Te ha resultado sencillo analizar tu proyecto con Dr. Scratch? (1= No; 2= Regular; 3= Sí)*
- Sentimientos tras resultados. *¿Cómo te has sentido al comprobar los resultados? (1= Mal; 2= Regular; 3= Bien)*
- Comprensibilidad de los resultados. *¿Entiendes la información de la página de resultados? (1= No; 2= Regular; 3= Sí)*
- Motivación de mejora. *Tras leer la información, ¿tienes ganas de probar algo nuevo? (1= No; 2= Regular; 3= Sí)*
- Suficiencia del *feedback. Utilizando la información de la página de resultados que has seleccionado, trata de mejorar tu proyecto añadiendo algo nuevo. ¿Son suficientes las ideas y trucos de la página de resultados para mejorar tu programa? (1= No; 2= Regular; 3= Sí)*

		Sencillez	Sentimientos	Comprensibilidad	Motivación	Feedback
Rho de Spearman	**Atractivo**	**,241***	,063	**,208***	**,195***	,172
	Sencillez		**,324****	,005	-,015	-,016
	Sentimientos			,071	-,086	,120
	Comprensibilidad				,005	**,294****
	Motivación					**,211***
*Correlación significativa p(r) < 0,05 ** Correlación muy significativa p(r) < 0,01						

Tabla 12. Correlaciones *Rho* de *Spearman* entre las respuestas de nivel ordinal

Tal como se muestra en la Tabla 12, se localizaron varias correlaciones significativas:

- El atractivo de la web se correlaciona positiva y significativamente, aunque con baja intensidad, con la percepción de sencillez del análisis, la comprensibilidad de los resultados y la subsecuente motivación de mejora.
- La percepción de sencillez del análisis correlaciona positiva y significativamente, aunque con una intensidad moderada, con buenos sentimientos tras recibir los resultados.
- La comprensibilidad de los resultados se correlaciona positiva y significativamente, aunque con una intensidad moderada, con la percepción de suficiencia del feedback.
- Por último, la motivación para mejorar está correlacionada significativa y positivamente, aunque con baja intensidad, con la percepción de suficiencia del feedback.

Conclusiones y trabajos futuros

Este artículo presenta Dr. Scratch, una herramienta web libre/de código abierto que permite analizar proyectos Scratch para asignar automáticamente una puntuación de PC y para detectar errores potenciales o malas prácticas de programación, con el objetivo de ayudar a los aprendices a desarrollar sus habilidades de programación y PC, así como asistir a los docentes en las tareas de evaluación.

Con el objetivo de medir la efectividad de Dr. Scratch como herramienta de soporte a jóvenes programadores, organizamos una serie de talleres en los que participaron 109 estudiantes de entre 10 y 14 años de 8 centros educativos que ya tenían experiencia previa programando con Scratch. Los alumnos analizaron uno de sus proyectos Scratch con Dr. Scratch, leyeron la información del informe de resultados ofrecido por la herramienta, trataron de mejorar sus programas siguiendo las instrucciones y, finalmente, analizaron de nuevo sus proyectos. Los resultados muestran que, de media, los alumnos mejoraron su puntuación de PC en 1,45 puntos, pasando de 12,00/21 a 13,45/21 puntos, lo que representa una mejora estadísticamente significativa. En este sentido, el tamaño global del efecto, d=0,47, indica un efecto moderado que, teniendo en cuenta que se generó durante un taller de tan solo una hora, subraya el impacto real que el uso de Dr. Scratch tuvo sobre las habilidades de programación y el desarrollo del PC de los participantes.

Los resultados muestran que el informe de retroalimentación ofrecido por Dr. Scratch fue especialmente útil para alumnos de secundaria con una puntuación inicial de PC media (*en desarrollo*). No obstante, la herramienta no parece ser tan útil para alumnos con una puntuación inicial alta (*competente*), al menos en el entorno investigado de un taller de una hora de duración. En el futuro desarrollaremos nuevas investigaciones para tratar de comprobar cómo mejorar el *feedback* que ofrece la herramienta. Además, nuevos estudios pueden ayudarnos a descubrir si las diferencias de rendimiento entre estudiantes de primaria y secundaria se deben a la propia herramienta o se encuentran relacionados con el desarrollo de la maduración meta-cognitiva

de los aprendices.

A corto plazo también tenemos planificadas nuevas investigaciones para tratar de encontrar correlaciones con otras herramientas que miden el PC de los alumnos, como el Test de Pensamiento Computacional (Román-González, 2015). De igual forma, se medirá la efectividad de la evaluación de Dr. Scratch comparando sus resultados con los de evaluadores expertos de primaria y secundaria. Estas investigaciones nos ayudarán a seguir ajustando y mejorando el funcionamiento del análisis de PC de Dr. Scratch.

En relación a las funcionalidades de Dr. Scratch, en el momento de escribir este artículo estamos trabajando en varias mejoras:

- Plug-ins para navegadores: con plug-ins para Firefox y Chrome, los aprendices podrán analizar sus proyectos mientras programan en el sitio web de Scratch.
- Cuentas de usuario: los alumnos podrán mantener un registro de sus análisis para estudiar su progresión en el tiempo.
- Cuentas de docente: los educadores podrán agrupar y seguir a sus alumnos, y mantener un registro de su evolución.
- Traducciones a nuevos idiomas: en este momento la herramienta está disponible en castellano e inglés, pero tenemos planificado incrementar el número de idiomas con la ayuda de la comunidad.
- *Gamificación* y funcionalidades de red social: se añadirán nuevas funcionalidades que permitirán a los usuarios comunicarse e intercambiar ideas y desafíos.
- 'App' para dispositivos móviles: estamos desarrollando una 'app' HTML5 para extender las posibilidades sociales y de *gamificación*.

Agradecimientos

El trabajo de todos los autores ha sido financiado parcialmente por la Comunidad de Madrid bajo el proyecto "eMadrid - Investigación y Desarrollo de tecnologías para el e-learning en la Comunidad de Madrid" (S2013/ICE-2715). Los autores están muy agradecidos a los docentes y alumnos de los colegios e institutos participantes en el estudio: Lope de Vega, Carles Salvador, Miguel de Cervantes, Villa Devoto, Camp de Túria, Mestre Ramón Esteve, Vicente Aleixandre and Fuente San Luis. También queremos agradecer a Eva Hu Garres and María Luz Aguado por su trabajo en el soporte técnico con Dr. Scratch.

Presentación del artículo: 31 de julio de 2015
Fecha de aprobación: 2 de septiembre de 2015
Fecha de publicación: 15 de septiembre de 2015

Moreno-León, J., Robles G., & Román-González, M. (2015). Dr. Scratch: Análisis Automático de Proyectos Scratch para Evaluar y Fomentar el Pensamiento Computacional. *RED, Revista de Educación a Distancia.46 (10)*. 15 de Septiembre de 2015. Consultado el (dd/mm/aaaa) en http://www.um.es/ead/red/46

Referencias

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. En 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*. Recuerado de 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)*