ReVisión
Revista de Investigación en Docencia Universitaria de la Informática
AENUI Asociación de Enseñantes Universitarios de la Informática

Programar para aprender en Educación Primaria y Secundaria: ¿qué indica la evidencia empírica sobre este enfoque?

Jesús Moreno-Léon,
Programamos
jesus.moreno@programamos.es

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

Marcos Román-González
U. Nacional de Educación a Distancia
mroman@edu.uned.es

Resumen

En los últimos años varios países europeos han introducido la programación informática y el pensamiento computacional en el currículum de Educación Primaria y Secundaria. En España asistimos, a su vez, a un debate sobre la necesidad de su inclusión y el enfoque a seguir para su puesta en práctica. Básicamente se pueden identificar dos aproximaciones: mientras algunas naciones han creado una asignatura específica centrada en el desarrollo de habilidades informáticas, otras han optado por usar estas habilidades como un instrumento para el aprendizaje de otras asignaturas. Estas diferencias se deben, al menos parcialmente, a la falta de evidencia científica sobre el desarrollo del pensamiento computacional a través de la programación en edad escolar y sobre sus posibilidades de transferencia para la adquisición de otras competencias. Por ello, consideramos fundamental aportar evidencia empírica basada en investigaciones desarrolladas con los recursos y bajo las circunstancias actuales de las aulas de nuestro país. Este artículo trata de presentar evidencias de investigaciones que muestran que el desarrollo del pensamiento computacional a través de la programación tiene un impacto positivo en el aprendizaje de distintas disciplinas, como las matemáticas, los idiomas, las ciencias o la narrativa. Así mismo, se discuten dos temas de especial relevancia en el ámbito educativo: las diferencias en la efectividad de la transferencia en función de la edad de los estudiantes y los requisitos de formación de los docentes para conseguir los objetivos marcados.

Palabras clave: Programación, pensamiento computacional, educación, primaria, evidencia empírica.

1. Introducción

En los últimos años han surgido todo tipo de iniciativas que persiguen que los estudiantes, desde edades tempranas, se acerquen al mundo de la programación informática y el pensamiento computacional [22]. Se pueden encontrar ejemplos de este tipo de iniciativas en todos los continentes, como en Estados Unidos,[1] Singapur,[2] Australia[3] o Nigeria [10]. En Europa varios países han comenzado a introducir contenidos de programación en su currículum nacional, tanto en Educación Primaria como Secundaria [1].

En este sentido se observan dos enfoques muy diferentes en lo relativo a los objetivos y metodologías con que estos contenidos se han introducido en el currículum de distintos países. Así, mientras naciones como Inglaterra o Eslovaquia han creado una asignatura específica centrada en el desarrollo de habilidades informáticas, otras como Estonia o Finlandia han optado por usar estas habilidades como un instrumento para el aprendizaje de otras asignaturas [3].

Esta falta de estandarización se debe, al menos parcialmente, a la escasez de evidencia científica sobre el desarrollo del pensamiento computacional a través de la programación en edad escolar y sobre sus posibilidades de transferencia para la adquisición de otras competencias, tal como ponen de manifiesto diversas revisiones del estado del arte que se han publicado recientemente [7, 11].

En este artículo se sintetizan y discuten las conclusiones e implicaciones de investigaciones actuales que se han desarrollado en nuestro país, en las que la programación se ha utilizado como un instrumento educativo con el objetivo de acelerar el aprendizaje de la asignatura en la cual se integra. De este modo tratamos da dar respuesta, al menos de forma tentativa,

[1] https://www.whitehouse.gov/blog/2016/01/30/computer-science-all
[2] https://portal.imda.gov.sg/Sub/Talent/Student-Programmes/Code-for-Fun
[3] http://www.australiancurriculum.edu.au/technologies/digital-technologies/curriculum/f-10

a las siguientes preguntas de investigación:

1. ¿En qué asignatura(s) se produce una mayor aceleración del aprendizaje cuando se integra la programación informática en la misma?
2. ¿Existen evidencias que muestren diferencias en la aceleración del aprendizaje en función de la edad de los estudiantes?
3. Adicionalmente, ¿influye el grado de experiencia en programación informática del profesor para conseguir que la integración en su asignatura acelere en mayor medida el aprendizaje de la misma?

2. Programar para aprender

La idea de usar la programación para aprender otras disciplinas no es nueva. Hace varias décadas Seymour Papert, quien participó en el desarrollo del lenguaje de programación Logo, explicaba en su libro Mindstorms que «al programar el ordenador, el niño adquiere un sentido de maestría frente a uno de los tipos de tecnología más potente y novedosa, y establece un contacto íntimo con algunas de las ideas más profundas de las ciencias, las matemáticas y la construcción de modelos intelectuales» [18].

Durante los años 70 y 80 se investigó en profundidad el impacto de la programación con Logo en el aprendizaje de otras asignaturas, como ponen de manifiesto varias revisiones de la literatura que se realizaron durante los 90 [4, 5, 17]. Si bien muchos de los estudios revisados ofrecían conclusiones prometedoras, también se publicaron trabajos que no observaron beneficios en el aprendizaje de los estudiantes tras las intervenciones.

No obstante, no existen apenas investigaciones empíricas sobre efectos de transferencia de la programación que hayan sido desarrolladas en el siglo XXI. Por tanto, es necesario «llenar este hueco aprendiendo del pasado y reviviendo este área de investigación» [20] usando los nuevos entornos de programación y herramientas que se han creado en los últimos años. La irrupción de los lenguajes visuales, como Scratch [19], puede suponer un salto cualitativo en el aprendizaje de la programación informática, puesto que estos lenguajes parecen superar las tres limitaciones básicas de los lenguajes textuales como Logo, a saber: las dificultades de sintaxis, la poca variedad de los productos de programación derivados, y lo solitario del contexto de aprendizaje. Lenguajes como Scratch, con sus características de *suelo bajo*, *techo alto* y *paredes anchas*, que están integrados en comunidades sociales de aprendizaje, estarían precisamente posibilitando trascender el enfoque de la programación como contenido, hacia una nueva perspectiva en donde la programación se constituye como herramienta para el desarrollo de distintas habilidades (de las cuales el pensamiento computacional sería la nuclear) y el aprendizaje de distintas asignaturas, tal y como revisamos en este artículo.

[4] http://ineverycrea.net

Dado que parece haber un salto cualitativo en las condiciones objetivas en las cuales los alumnos de hoy en día se relacionan con los lenguajes de programación, debe realizarse en paralelo una nueva oleada de investigación que compruebe si los antiguos resultados se replican y, además, aporte evidencia sobre emergentes preguntas.

A pesar de que los autores han realizado una revisión de investigaciones llevadas a cabo con este enfoque desde el año 2007 [14], para este artículo se ha decidido limitar la discusión a los estudios que se han desarrollado en España, puesto que las circunstancias de las aulas en diferentes países no son siempre equivalentes.

3. Investigaciones desarrolladas recientemente en nuestro país

3.1. Diferencias en términos de transferencia en función de la asignatura en la que se integra la programación

Durante el curso 2014/15, en colaboración con la iniciativa Inevery Crea,[4] desarrollamos una investigación con varios colegios con el objetivo de estudiar diferencias en términos de transferencia de las habilidades de programación y pensamiento computacional de los estudiantes para el aprendizaje de diferentes asignaturas [16].

Para ello, los docentes participantes recibieron un curso de formación de cuatro semanas con el lenguaje de programación Scratch. A continuación, los docentes establecieron dos grupos de estudiantes, uno de control y otro experimental, que estaban formados por alumnos de características similares. Los estudiantes de ambos grupos realizaron una prueba inicial con el objetivo de medir los conocimientos del tema que iban a comenzar a trabajar en la asignatura correspondiente. A partir de ese momento, los estudiantes del grupo experimental comenzaron a trabajar la asignatura a través de actividades de programación, mientras el grupo de control lo hizo con actividades *tradicionales*, es decir, siguiendo la misma metodología y utilizando los mismos recursos que hasta ese momento. Al finalizar la unidad, ambos grupos de estudiantes realizaron una prueba final de conocimientos de la asignatura. Además, los estudiantes contestaron un cuestionario, y se recogieron los proyectos Scratch desarrollados en el grupo experimental, que fueron analizados con la herramienta Dr. Scratch [15].

El mismo experimento se realizó en dos colegios diferentes: en uno de los centros se desarrolló en 6º de Primaria en la asignatura de Matemáticas, en una unidad relativa a los ángulos, mientras que en el otro colegio se hizo en 6º de Primaria en la asignatura de Ciencias Sociales, en un tema relacionado con la Unión Europea. La figura 1, que muestra una captura de un proyecto Scratch programado por los estudiantes durante la intervención, puede servir para ilustrar el tipo de actividades que desarrollaron los grupos experimentales. El proyecto

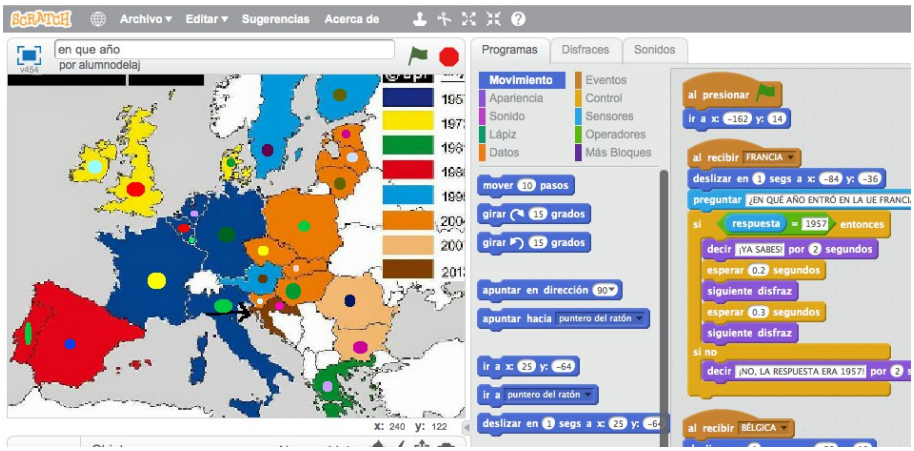


Figura 1: Captura de *¿En qué año?*. Proyecto Scratch desarrollado en el grupo experimental de 6º de Primaria en la clase de Ciencias Sociales. Disponible en `https://scratch.mit.edu/projects/53945786/`.

mostrado consiste en un concurso interactivo de preguntas y respuestas en el que el usuario es preguntado por la fecha de adhesión de los diferentes países miembro de la Unión Europa.

En ambos casos al comparar el aprendizaje de los grupos de control y experimental, los resultados muestran que el uso de la programación aceleró la curva de aprendizaje, tal como puede verse en la figura 2. Sin embargo, el tamaño del efecto fue dos veces mayor en la asignatura de Ciencias Sociales que en la de Matemáticas. Al analizar los proyectos Scratch creados por los alumnos, se comprueba que, en términos de pensamiento computacional, la integración de la programación en Matemáticas fue más dura cognitivamente. En la asignatura de Ciencias Sociales, de acuerdo con las respuestas de los cuestionarios, el hecho de que la demanda cognitiva fuera menor se tradujo en que los estudiantes disfrutaron más y mostraron una mayor autonomía y motivación que los estudiantes de Matemáticas.

3.2. Diferencias en términos de transferencia en función de la edad de los aprendices

La misma intervención descrita en la sección anterior se realizó también en un tercer colegio, pero esta vez en 2º de Primaria en la asignatura de Lengua, en el ámbito de una unidad en la que se trabajaba la estructura narrativa [16].

Mientras que, tal como se ha explicado, en los dos grupos de 6º Primaria la inclusión de la programación aceleró la curva de aprendizaje, este no fue el caso en 2º Primaria, aunque es importante señalar que tampoco la deceleró, puesto que, tal como puede comprobarse en la figura 3, no se detectaron diferencias significativas entre los resultados del grupo de control y el experimental.

Estos resultados, que muestran diferencias en términos de transferencia entre estudiantes de últimos cursos de Primaria, que son adolescentes tempranos, y alumnos de primeros cursos de Primaria, que se encuentran aún en la etapa de la infancia, van en línea con otras investigaciones que comparan las posibilidades de transferencia en función de la madurez cognitiva de los aprendices en otros campos [21]. Así pues, la evidencia empírica acumulada hasta ahora indica que no cabe esperar transferencia del aprendizaje realizado mediante la programación informática a los contenidos propios de las asignaturas hasta el último ciclo de Educación Primaria.

3.3. Diferencias en términos de transferencia en función de la formación de los docentes

Una investigación similar se desarrolló en el curso 2013/14 trabajando con grupos de 4º y 5º de Primaria en la clase de Inglés [13]. En esta ocasión, en la investigación participaron dos docentes: uno que tenía formación previa en programación y otro que no disponía de estos conocimientos, aunque recibió un curso de formación de 3 horas.

Los resultados obtenidos por ambos docentes son muy diferentes, tal como se muestra en la figura 4. En el caso del docente que tenía conocimientos previos de programación, el grupo experimental, que trabajó la unidad correspondiente usando actividades de programación con Scratch, obtuvo unos resultados significativamente mejores que el grupo de control. Sin embargo, los resultados no fueron los mismos en el caso del docente sin conocimientos previos. De hecho, en ese caso los estudiantes del grupo de control, que trabajaron la unidad de forma *tradicional*, aprendieron más que sus compañeros del grupo experimental.

Estos resultados parecen ser consistentes con los descubrimientos de otros estudios [12] que indican que contar con docentes bien formados es esencial para optimizar el aprendizaje de los estudiantes y para conseguir los objetivos esperados.

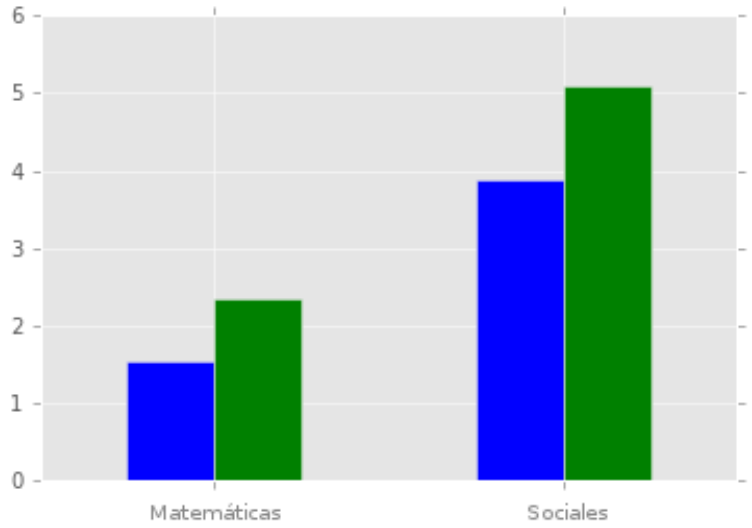


Figura 2: Mejora experimentada entre la prueba inicial y final: a la izquierda, los resultados para la clase de Matemáticas; a la derecha, los de la clase de Ciencias Sociales. Los grupos de control se muestran en color azul; los grupos expermientales en color verde.

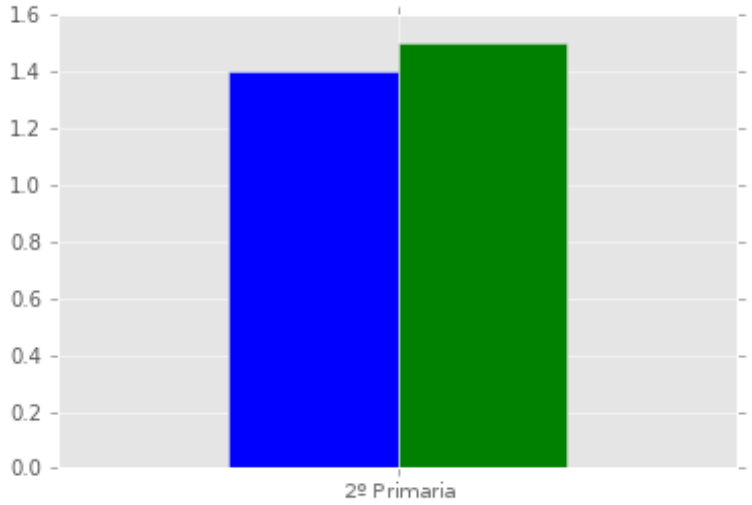


Figura 3: Mejora experimentada entre la prueba inicial y final: resultados para la clase de 2º de Primaria. El grupo de control se muestra en color azul; el grupo expermiental en color verde.

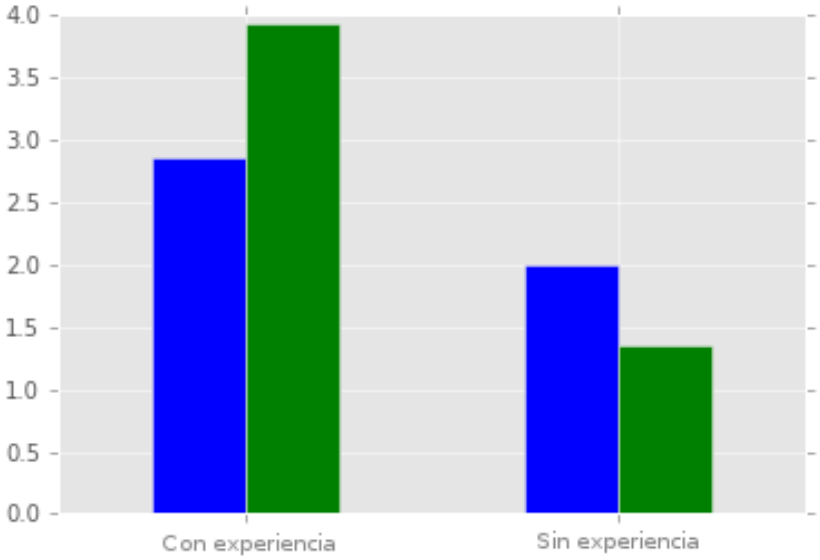


Figura 4: Mejora experimentada entre la prueba inicial y final: a la izquierda, los resultados para la clase del docente con experiencia previa en programación; a la derecha, los de la clase del docente sin experiencia previa. Los grupos de control se muestran en color azul; los grupos expermientales en color verde.

4. Discusión

En un momento en el que en España se discute la necesidad de introducir la programación en el currículum nacional y se debate sobre el enfoque a seguir para su enseñanza, los resultados de estas investigaciones pueden ofrecer algunas conclusiones de utilidad para los responsables educativos.

En primer lugar, de cara a decidir el curso en el que comenzar a introducir la programación y el pensamiento computacional, hay que tener en cuenta que parecen existir diferencias en la efectividad de la transferencia de estas habilidades a otros dominios en función de la edad de los estudiantes. Alumnos de los últimos cursos de Primaria, que se acercan a la adolescencia, tienen unos niveles de madurez cognitiva que pueden aumentar las posibilidades de transferencia. No obstante, hay que recalcar que aunque el uso de la programación no implique una mejora de rendimiento académico en los primeros cursos de Primaria, tampoco se produce una desaceleración, por lo que podría ser interesante introducir este recurso a esta edad. De este modo los estudiantes podrían comenzar a desarrollar su pensamiento computacional sin comprometer sus resultados académicos.

Por otra parte, parece adecuado comenzar a usar la programación como herramienta transversal en asignaturas en las que la integración sea sencilla, como las Ciencias Sociales, de forma que los estudiantes puedan coger confianza y autonomía. En una fase posterior, introducir estas actividades en asignaturas más complejas, como las Matemáticas, puede resultar más fácil y se pueden obtener mejores resultados.

Por último, parece evidente que la formación del profesorado tiene un impacto muy importante en los resultados de aprendizaje que obtengan los estudiantes. En consecuencia, una pregunta recurrente en todos los países que han comenzado a implantar estas enseñanzas es cómo conseguir suficientes docentes bien formados para enseñar a través de la programación y el pensamiento computacional para todas las escuelas. Este problema se acentúa por la alta demanda de profesionales en el campo de la informática que ya se está produciendo en muchos países y que parece que va a sobrepasar claramente a la oferta en los próximos años, también en Europa [6], lo que contribuye a acelerar la pérdida de especialistas en las aulas [2]. De hecho, según estimaciones de Code.org los docentes especialistas en informática tan solo estarán en las escuelas durante 3 años antes de cambiar a otro campo profesional.[5] ¿Cómo lograr docentes bien formados en informática de un modo sostenible, por tanto?

En este sentido coincidimos con la visión de Mark Guzdial [8], y pensamos que quizás el único camino viable pasa por la contratación urgente de especialistas en la enseñanza y el aprendizaje de la informática en las Facultades de Educación, de forma que estos especialistas puedan liderar durante las próximas décadas departamentos encargados de la docencia tanto para la actualización de maestros y profesores en activo, como especialmente para la formación de estudiantes de magisterio y máster del profesorado. En nuestro país este paso debe realizarse de forma inmediata, puesto que otras naciones, como Alemania o Austria [9], han comenzado a invertir en esta línea y están contratando a los especialistas con los que se

[5] Véase `https://code.org/about/evaluation/summary`.

cuenta en Europa, también españoles, por lo que pronto se producirá una escasez importante de talento en este campo.

5. Conclusiones

En este trabajo se resumen los resultados de recientes investigaciones desarrolladas con los recursos y bajo las circunstancias actuales de las aulas españolas en relación al uso de la programación y el pensamiento computacional como recurso para el aprendizaje de diversas asignaturas y competencias.

En un momento en el que en nuestro país se discute la necesidad de introducir la programación en el currículum nacional y se debate sobre el enfoque a seguir para su enseñanza, las implicaciones de este tipo de estudios pueden resultar de interés para las administraciones públicas y entidades privadas que están participando en la elaboración de la próxima ley educativa.

Este artículo pone de manifiesto que es necesario realizar más investigaciones que puedan utilizarse para comprobar qué estrategias funcionan y qué enfoques no lo hacen cuando se usa la programación en el aula desde edades tempranas. Si la comunidad educativa y científica quieren ayudar a los responsables políticos en la toma decisiones en este campo, se requiere desarrollar un mayor número de estudios empíricos rigurosos que analicen de forma crítica el rol de la introducción de la informática en las escuelas.

Agradecimientos

Este trabajo 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).

Referencias

[1] Anja Balanskat y Katja Engelhardt. Computing our future: Computer programming and coding. Priorities, school curricula and initiatives across Europe. Technical report, European Schoolnet, 2015.

[2] D Bernier y J Margolis. The revolving door: Computer science for all and the challenge of teacher retention. Technical report, Exploring Computer Science, 2014.

[3] Stefania Bocconi, Augusto Chioccariello, Giuliana Dettori, Anusca Ferrari, Katja Engelhardt, Panagiotis Kampylis y Yves Punie. Developing computational thinking in compulsory education - Implications for policy and practice. Technical report, Publications Office of the European Union, 2016.

[4] Douglas H Clements y Julie Sarama Meredith. Research on logo: Effects and efficacy. *Journal of Computing in Childhood Education*, 4(4):263–290, 1993.

[5] Douglas H Clements y Julie Sarama. Research on logo: A decade of progress. *Computers in the Schools*, 14(1-2):9–46, 1997.

[6] Karsten Gareis, Tobias Hüsing, Strahil Birov, Inna Bludova, Carola Schulz y Werner Korte. e-Skills for jobs in Europe: measuring progress and moving ahead. Technical report, European Commission, 2014.

[7] Shuchi Grover y Roy Pea. Computational thinking in k–12. A review of the state of the field. *Educational Researcher*, 42(1):38–43, 2013.

[8] Mark Guzdial. The bottleneck in increasing accessibility to CS education is producing enough CS teachers, 2015. BLOG@CACM. Available at `http://cacm.acm.org/blogs/blog-cacm/192586-the-bottleneck-in-increasing-accessibility-to-cs-education-is-producing-enough-cs-teachers`, accessed 2017.

[9] Mark Guzdial. Growing CS Ed through Schools of Ed: Report from Oldenburg, 2015. Computing Education Blog. Available at `https://computinged.wordpress.com/2015/07/13/growing-cs-ed-through-schools-of-ed-report-from-oldenburg/`, accessed 2017.

[10] Charity O. Igbokwe. Recent curriculum reforms at the basic education level in Nigeria aimed at catching them young to create change. *American Journal of Educational Research*, 3(1):31–37, 2015.

[11] Computing in Schools Special Interest Network. Computing and digital literacy: Call for a holistic approach. Technical report, Council of European Professional Informatics Societies, 2015.

[12] Orni Meerbaum-Salant, Michal Armoni y Mordechai Ben-Ari. Learning computer science concepts with Scratch. *Computer Science Education*, 23(3):239–264, 2013.

[13] J Moreno-León y Gregorio Robles. Computer programming as an educational tool in the english classroom a preliminary study. In *Global Engineering Education Conference (EDUCON), 2015 IEEE*, pages 961–966. IEEE, 2015.

[14] Jesús Moreno-León y Gregorio Robles. Code to learn with Scratch? a systematic literature review. In *Global Engineering Education Conference (EDUCON), 2016 IEEE*, pages 150–156. IEEE, 2016.

[15] Jesús Moreno-León, Gregorio Robles y Marcos Román-González. Dr. Scratch: automatic analysis of Scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia*, 15(46), 2015.

[16] Jesús Moreno-León, Gregorio Robles y Marcos Román-González. Code to learn: Where does it belong in the k-12 curriculum? *Journal of Information Technology Education: Research*, 15:283–303, 2016.

[17] David B Palumbo. Programming language/problem-solving research: A review of relevant issues. *Review of educational research*, 60(1):65–89, 1990.

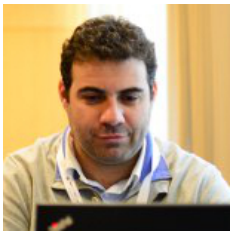
[18] Seymour Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., 1980.

[19] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[20] Ronny Scherer. Learning from the past - the need for empirical evidence on the transfer effects of computer programming skills. *Frontiers in Psychology*, 7:1390, 2016.

[21] Rachel Schiff y Eli Vakil. Age differences in cognitive skill learning, retention and transfer: The case of the Tower of Hanoi puzzle. *Learning and Individual Differences*, 39:164–171, 2015.

[22] Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.



Jesús Moreno León. Tras más de 10 años como docente de secundaria y formación profesional, y tras su trabajo como asesor técnico en el Instituto Nacional de Tecnologías Educativas y Formación del Profesorado, Jesús se encuentra volcado en la investigación y promoción del pensamiento computacional desde edades tempranas. Para ello compagina su labor al frente de la organización sin ánimo de lucro Programamos con el desarrollo de su tesis doctoral en el grupo KGBL3 de la Universidad Rey Juan Carlos.

Gregorio Robles es profesor en la Universidad Rey Juan Carlos. Obtuvo su doctorado en 2006 con una tesis sobre investigación empírica en ingeniería del software libre/de código abierto. Además de esta línea de investigación, Gregorio también es muy activo en el campo de las tecnologías educativas, especialmente en lo relativo al estudio y evaluación del pensamiento computacional.

Marcos Román González, doctor en educación, es profesor en la Universidad Nacional de Educación a Distancia. Sus líneas de investigación principales están relacionadas con el pensamiento computacional, la código-alfabetización y la enseñanza de la informática, con un énfasis especial en su evaluación. Sus publicaciones más recientes incluyen un artículo en Computers & Education sobre la integración transversal de lenguajes de programación visuales en la Educación Primaria.