# Propuesta de Metodología Basada en Metáforas para la Enseñanza de la Programación a Niños

Diana Pérez-Marín, Raquel Hijón-Neira, Mercedes Martín-Lope

CÓMO REFERENCIAR ESTE ARTÍCULO:

D. Pérez-Marín, R. Hijón-Neira, M. Martín-Lope. ''A Methodology Proposal based on Metaphors to teach Programming to children'', 2018
DOI: https://doi.org/10.1109/RITA.2018.2809944

***Title*— A Methodology Proposal based on Metaphors to teach Programming to children**

***Abstract*— Interest in studying Computer Science has been extended worldwide to children. However, it is unclear which educational method should be used. Teachers need some guides to approaching this task. Therefore, this paper proposes using metaphors such as recipe/program (and sequence), pantry/ memory, and boxes /variables. It also illustrates the possibility of applying these metaphors to any resource available to the teacher. Four step-by-step scripts of how to use the metaphors in class are provided, with the opinions of 62 children (enrolled in 4th, 5th and 6th Primary courses, 9 to 11 years in age) and their teacher's opinion.**

***Index Terms*—Computer Science Education, metaphor, Computational thinking.**

## I. Introducción

Saber programar es una de las competencias clave que sería deseable que pudiesen adquirir todos los estudiantes en el siglo XXI [1-3]. Sin embargo, cuando se revisa la literatura de las experiencias realizadas, para investigar si efectivamente los estudiantes adquieren esta competencia, muchos estudiantes universitarios no lo han conseguido. Por el contrario, les cuesta descomponer problemas, desarrollar algoritmos, e implementar estos algoritmos usando algún lenguaje de programación [4].

En los últimos años, el interés en estudiar Programación se ha extendido a niveles inferiores de enseñanza. Esto puede deberse a que algunos estudios sugieren que, si se enseñase programación en edades más tempranas, se podría desarrollar el pensamiento computacional necesario para poder posteriormente crear programas, resolviendo el problema detectado anteriormente, e incluso poder resolver en general problemas que de otra forma no podrían [5].

Muchos estudios se centran en la enseñanza de la programación para niños mediante el uso de Scratch [5-8]. Sin embargo, no hay todavía artículos que proporcionen una metodología de enseñanza de la Programación que permita desarrollar el pensamiento computacional deseado en estos niveles, ni que apoye en este objetivo a los docentes sin conocimientos previos en Informática.

Por el contrario, se encuentran dificultades enseñando conceptos básicos como programa [9], bucles [10], estructuras de control y algoritmos [11]. Las dificultades en muchos casos se pueden deber a la falta de una metodología de enseñanza de estos conceptos en Educación Primaria [9-10]. Se hace patente la necesidad de que los profesores sin conocimientos previos de Informática dispongan de alguna guía para llevar a cabo esta nueva tarea de enseñanza de la programación y el fomento del pensamiento computacional en edades tempranas [5,8,14].

En este artículo, se propone por primera vez el uso de metáforas para introducir a los niños en los conceptos básicos de programación. En particular, se proponen 4 grupos de metáforas: M1, que engloba las metáforas que relacionan programar con cocinar, programa-secuencia con receta, memoria con despensa y variables con cajas; M2, que engloba las metáforas de entrada-salida que relacionan la entrada con el teclado y la salida con la pantalla; M3, que engloba las metáforas de condicionales que relacionan la posibilidad de los niños de tomar decisiones como los ordenadores también pueden tomar decisiones; y M4, que engloba las metáforas de bucles que relacionan la posibilidad de poner la mesa para X personas repitiendo las acciones como los ordenadores pueden repetir las instrucciones varias veces. Se proporcionan también cuatro guías detalladas de cómo poner en la práctica estas metáforas en clase, independientemente de los recursos de los que disponga el profesor.

La razón por la que se propone el uso de metáforas es su utilidad probada como herramienta educativa cuando el dominio de enseñanza es abstracto. Las metáforas se centran en los conceptos, y proporcionan a los estudiantes una mejor organización de las ideas, y una forma de pensar más directa.

Además, usar metáforas no necesita un software ni un hardware especial, tan solo el lenguaje que permita a los profesores convertir conceptos difíciles y abstractos en ideas más simples y fáciles de adquirir. Esto es, justo lo que se quiere conseguir con el pensamiento computacional, una forma de pensar clara y directa, y que permita a los estudiantes desarrollar con éxito la competencia de programar y resolver problemas [15].

Sin embargo, las metáforas también deben tratarse con cuidado. En usos previos de enseñanza basada en metáforas

Manuscrito recibido el 19 de mayo de 2017. Revisado 13 de junio. Aceptado 30 de noviembre.

English versión received January, 12th, 2018. Revised February, 6th. Accepted February, 16th.

D. Pérez-Marín y R. Hijón-Neira trabajan en el campus de Móstoles de la Universidad Rey Juan Carlos, Madrid, España (diana.perez@urjc.es, raquel.hijon@urjc.es).
(https://orcid.org/0000-0003-3390-0251)
(https://orcid.org/0000-0003-3833-4228)
M. Martín-Lope trabaja en el campus de Madrid de la Universidad Rey Juan Carlos, Madrid, España (mercedes.martin@urjc.es).
(https://orcid.org/0000-0002-4442-8215)

en Informática en niveles universitarios se ha advertido que no se debe llegar a "romper" la metáfora [16]. Esto es, hay que usar cada metáfora en su ámbito de acción y cuidar el lenguaje, por este motivo se proporcionan las guías detalladas a los profesores, y es objeto de investigación en este artículo cómo se puede cuidar el lenguaje para que los estudiantes puedan comprender y considerar de utilidad estas metáforas, y los profesores las vean también de utilidad y las quieran aplicar en sus clases. Para validar esta propuesta se proporcionan los resultados de un estudio piloto realizado en un colegio de la Comunidad de Madrid en España.

El objetivo principal de este primer estudio piloto es validar las guías desarrolladas con las metáforas propuestas. 62 niños de 4º, 5º y 6º de Educación Primaria (de 9 a 11 años de edad) y su profesor evaluaron las guías y nos expresaron su opinión en un cuestionario que se les proporcionó.

Tanto los estudiantes como el profesor validaron la utilidad de las metáforas y mostraron su actitud favorable a usarlas en clase. Los indicadores utilizados fueron el porcentaje de estudiantes que encontraron útiles las metáforas, el porcentaje de estudiantes que encontraron difíciles las metáforas, y el porcentaje de estudiantes que solicitaron una forma alternativa de enseñanza de los conceptos informáticos.

El artículo se organiza de la siguiente forma: la Sección 2 revisa el estado del arte de los métodos para enseñar conceptos básicos de programación para desarrollar el pensamiento computacional en los niños; la Sección 3 proporciona las guías que usan por primera vez las metáforas para la enseñanza en Educación Primaria de programación; la Sección 4 proporciona los resultados del estudio piloto realizado; y finalmente, la Sección 5 termina el artículo con la conclusión.

## II. Revisión del Estado del Arte

### A. *Interés mundial en enseñar programación a los niños*

Heintz et al. presentaron una revisión de cómo Australia, Reino Unido, Estonia, Finlandia, Nueva Zelanda, Noruega, Suecia, Corea del Sur, Polonia y Estados Unidos enseñan programación en Informática en su Educación Primaria y Secundaria [15]. El estudio revela que normalmente la programación impartida en la asignatura de Informática es obligatoria en Educación Primaria y optativa en Educación Secundaria. La Tabla I recoge la situación particular de cada país, con celdas en blanco indicando la falta de información en esos casos. En todo caso, la información recogida pone de relieve el interés mundial en enseñar programación desde edades tempranas.

La tendencia general es ir fomentando desde edades tempranas el pensamiento computacional y la competencia digital. En Educación Secundaria se ofrecen cursos más diversos de Informática y su impacto en la sociedad [15].

En España, la asignatura donde se incluye la enseñanza de la Programación es de Libre Configuración Autonómica. Esto significa que las distintas Comunidades Autónomas pueden decidir si es obligatoria en cada caso.

Por ejemplo, en la Comunidad de Madrid y para la etapa de Educación Primaria, la asignatura se establece como de oferta obligada y se denomina "Tecnología y Recursos Digitales para la mejora del aprendizaje".

TABLA I
INTERÉS MUNDIAL EN LA ENSEÑANZA DE INFORMÁTICA EN EDUCACIÓN PRIMARIA Y SECUNDARIA [12]

| País | Ed. Primaria | Ed. Secundaria |
|---|---|---|
| Australia | Obligatoria | Obligatoria |
| Inglaterra | Obligatoria | |
| Estonia | Obligatoria | Obligatoria |
| Finlandia | Obligatoria | |
| N. Zelanda | | Optativa |
| Noruega | | Optativa |
| Suecia | Obligatoria | Optativa |
| Korea Sur | Obligatoria | Optativa |
| EE.UU. | | Optativa |
| Macedonia | Obligatoria | |

Esta asignatura tiene un contenido marcado por la ley para toda Educación Primaria, aunque no se establece qué se debe enseñar en cada curso. Se suele impartir a través de las distintas áreas que conforman el currículo (cada colegio puede decidir cómo organizarlo).

En el currículo oficial también se regula el desarrollo la competencia digital como una de las ocho competencias clave que todos los estudiantes deben adquirir al finalizar la etapa de Educación Secundaria obligatoria [17].

### B. *Enfoques para Enseñar Programación a los Niños*

Una de las herramientas más utilizadas a nivel mundial para enseñar programación a niños es el entorno de programación visual **Scratch** [18]. El objetivo es que, mientras interactúan con Scratch, los estudiantes puedan aprender conceptos básicos como secuencia, bucle, paralelismo, eventos, condicionales, operadores y datos [7,19]. Así, por ejemplo, el concepto de secuencia se puede programar moviendo una distancia corta un gato y un perro como se muestra en la Figura 1.

El concepto de bucle, como mecanismo para ejecutar la misma secuencia varias veces, se podría enseñar con el comando *repite* seguido del número de iteraciones que se desean. El concepto de condicional, comprendido como la habilidad para tomar decisiones según ciertas condiciones, se podría enseñar con el comando *si* que determina la visibilidad de los objetos. El concepto de dato, para guardar, recuperar y actualizar valores, se podría enseñar con variables que pueden guardar números o cadenas, y listas que pueden guardar una colección de números o cadenas.

Otros enfoques para la enseñanza de Programación y el desarrollo de su pensamiento computacional incluyen crear sus propios programas [6], usar Lego WeDo o robots Mindstorms EV3 [20], y enfoques sin tecnología, para

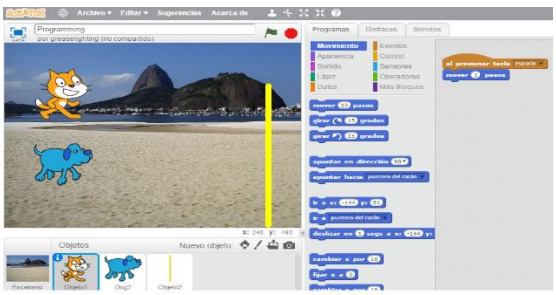


Fig. 1. Concepto de secuencia en Scratch

países en los que aunque consideran que la enseñanza de la Programación es importante, no tienen suficientes ordenadores, o profesores con conocimientos en la materia [5].

En estos enfoques, se enseña Informática mediante el uso de cuentos. Sin embargo, estos enfoques aún no han sido suficientemente evaluados para poder tener resultados significativos de su impacto [21].

### C. Uso de Metáforas

El lenguaje metafórico se usa todos los días, y se considera una competencia fundamental de nuestra forma de pensar [22]. Las metáforas conceptuales, esto es, mecanismos que proyectan de un dominio fuente a un dominio destino para facilitar la enseñanza de algún concepto abstracto, pueden ser valiosas herramientas educativas [16].

Las metáforas se han usado para enseñar Biología [23], Química [24], y Matemáticas [25]. El uso de metáforas para enseñar conceptos informáticos a nivel universitario también ha sido objetivo de interés [16,26]. Hay estudios que proponen metáforas concretas para enseñar conceptos abstractos como memoria dinámica [27], o matrices para el manejo de eventos en JAVA [28]. Sin embargo, no se encuentra en la literatura ejemplos del uso de metáforas para enseñar conceptos básicos de programación en Educación Primaria.

## III. Propuesta: Uso de Metáforas para Enseñar Programación

A partir de la revisión del estado del arte se puede afirmar que todavía no se tiene claro cómo enseñar conceptos básicos de programación a los niños. Los profesores están desorientados, carecen en muchos casos de conocimientos de Informática, y adolecen de la formación adecuada para enseñar programación a los niños [14]. En otros casos, los profesores no tienen demasiados recursos. Sólo tienen la pizarra y quizás un ordenador compartido por varios estudiantes [5].

Como se ha visto en la Sección II.C, las metáforas pueden ser una herramienta educativa poderosa si se usa con cuidado. Se conocen varios casos de éxito a nivel universitario, pero no hay estudios sobre su uso con niños en la literatura. Por otro lado, se pide más investigación sobre este tema [16]. Además, se ha enfatizado la importancia de guiar a los niños paso a paso en su aprendizaje [29].

TABLA II
RESUMEN DE LAS METÁFORAS USADAS EN LOS GUIONES

| ID | Guión | Concepto | Metafora |
|---|---|---|---|
| **M1** | Programa, programación, secuencia, memoria | Programa-secuencia | receta |
| | | Memoria, variables | Despensa, caja |
| **M2** | Instrucciones de entrada / salida | Salida | Pantalla Ordenador |
| | | Entrada | Teclado, reflejo en la pantalla PC |
| **M3** | Condicionales | Niño tomando decisiones | Ordenador tomando decisiones |
| **M4** | Bucles | Tu poniendo la mesa para X personas | Ordenador repitiendo instrucciones X veces |

Por todo esto, proponemos en este artículo, por primera vez, cuatro guiones basados en el uso de cuatro grupos de metáforas como una metodología para enseñar conceptos básicos de programación a niños y desarrollar su pensamiento computacional. Estos cuatro guiones engloban seis metáforas que están resumidas en la Tabla II. Además, se ofrece en la Tabla III una temporalización basada en nuestra experiencia utilizando los guiones. La información está clasificada por el curso en el que el estudiante está matriculado. Por ejemplo, 4º, 5º o 6º, entre 9 y 11 años, ya que a partir de esa edad se espera que los niños tengan la adecuada capacidad cognitiva para aprender metáforas [30].

Dependiendo de si los profesores disponen de equipos de visualización, con una aplicación que proyecte las metáforas, o una presentación de Powerpoint que las incluya. En ese caso, el tiempo requerido se reduce porque las metáforas, dibujos y pseudocódigos ya están escritos. De igual forma, si los profesores tienen Scratch podrían usar las metáforas al explicar cada uno de los conceptos de programación y utilizar los bloques visuales del programa para hacer ejercicios prácticos en cada caso. En cualquier caso, los profesores que no tengan la posibilidad de tener pantallas digitales pueden usar una pizarra o similar.

El proceso de explicación en todos los casos se divide en cuatro pasos sucesivos que reflejan el curso normal de introducción a la programación: (1) concepto de programa, programar, secuencia, memoria y variable; (2) instrucciones de entrada y salida (3), condicionales, y finalmente (4) bucles.

Para resaltar cuando se utiliza una metáfora en un guion aparecerá subrayada. Cuando se introduce un nuevo concepto aparecerá en negrita. Las variables aparecerán en cursiva. En algunos casos, se ofrecen diálogos de ejemplo entre profesores (P) y estudiantes (S).

### A. Guion para introducir a los niños a los primeros conceptos de programación

(P): “¿Has utilizado alguna vez un juego de ordenador o programa? ¿Conoces Youtube? Es un programa para ver videos. ¿y Paint? Es un programa para dibujar; ¿y Word? Es un programa que usas para escribir cartas, fichas, lo que

TABLA III
TIEMPO SUGERIDO PARA ENSEÑAR CONCEPTOS BÁSICOS DE PROGRAMACIÓN

| Nº | Conceptos | Curso | Con Programa | Con Pizarra |
|---|---|---|---|---|
| **1** | Programa, programación, secuencia, memoria | 6 | 1 h | 2 h |
| | | 5 | 1 h 12’ | 2 h 24’ |
| | | 4 | 1 h 24’ | 2 h 48’ |
| **2** | Instrucciones entrada / salida | 6 | 1 h | 2 h |
| | | 5 | 1 h 12’ | 2 h 24’ |
| | | 4 | 1 h 24’ | 2 h 48’ |
| **3** | Condicionales | 6 | 1 h | 2 h |
| | | 5 | 1 h 12’ | 2 h 24’ |
| | | 4 | 1 h 24’ | 2 h 48’ |
| **4** | Bucles | 6 | 1 h | 2 h |
| | | 5 | 1 h 12’ | 2 h 24’ |
| | | 4 | 1 h 24’ | 2 h 48’ |

quieras; vamos a ver sobre Google, ¿sabes lo que es Google? Es un programa para buscar en Internet.
Los alumnos pueden contestar que conocen alguno de estos programas, y en algunos casos todos ellos, pero lo principal es enseñarles que son programas de ordenador, y que pueden estar instalados en sus ordenadores.

(P): "¿Sabéis cómo funcionan los programas? Como una receta, como las que veis en Master Chef Junior por ejemplo"

"Si queréis cocinar una tortilla necesitáis herramientas y alimentos para hacerlo, así: una sartén, aceite de oliva, dos huevos y una pizca de sal"

"Después especificáis los pasos que tenéis que seguir, uno después del otro (para introducir el concepto de secuencia)… como cuando estáis en fila:

1. Pon la sartén en el fuego, 2. Pon aceite, 3. Pon dos huevos en una taza, 4. Bátelos, 5. Échalos en la sartén, 6. Bátelos hasta que estén batidos

"Un programa funciona así, como una secuencia de pasos, y en cada paso se ejecuta una instrucción"

"Vamos a ver un ejemplo: quiero escribir en la pantalla del ordenador un mensaje al usuario (tú eres el usuario, la persona que usa el ordenador)".

La Figura 2 representa la pizarra en ese momento; en el programa el profesor escribirá la instrucción en el lado izquierdo de la Figura 2, y en el lado derecho el recurso disponible; si el recurso es una pizarra el profesor dibujará la ejecución paso a paso; si es una pantalla, se mostrará la ejecución dinámicamente en el programa elegido.

(P): "En mi programa escribiré la instrucción: Escribe_en_pantalla (texto que quiero que salga), y aparecerá en la pantalla" como en la Figura 3.

Después para introducir el concepto de variable y memoria la metáfora de que programar es como cocinar continua:

(P): "pero el ordenador tiene un almacén de comida o despensa donde almacena todo lo que necesita, y lo va modificando para adaptarlo a sus necesidades. Esa despensa es la memoria del ordenador que permite almacenar variables como cajitas. Las variables son cajitas que puedes rellenar con lo que necesites. Por ejemplo (Figura 3): en la despensa tengo una taza donde pongo huevos, o harina, o azúcar,…, todo lo que se necesita para seguir mi receta; lo mismo en el ordenador donde guardo los números, o nombres, o mensajes que necesito mostrar al usuario".

| Programa (instrucción) | Pantalla (salida) |
|---|---|
| escribe_en_pantalla ("Hello") |  |
| Escribe_en_pantalla ("How are you?") |  |
| escribe_en_pantalla ("Hello")<br>escribe_en_pantalla ("How are you?") |  |

Fig. 2. Ejemplos de instrucciones de salida escritas en un programa; izquierda: instrucción de programa, derecha: salida en la pantalla

| Despensa | | Memoria | |
|---|---|---|---|
| Dibujo una taza |  | Dibujo una caja (variable) |  |
| La lleno de huevos |  | Escribo "Hello" dentro de ella |  |
| La vacío y la relleno de harina |  | La borro y escribo "Bye" en ella |  |
| No la vacío y le añado chocolate. Ahora tengo una taza con harina y chocolate |  | No la borro y le añado "have a nice day". Ahora tengo en la cajitaVariable "bye, have a nice day" |  |

Fig. 3. Ejemplos para ilustrar la metáfora de memoria-variables como despensa-caja (grupo de metáforas M1)

### *B. Guion para introducir a los niños a las instrucciones de entrada/Salida*

Vamos a combinar los conceptos: programa, memoria y pantalla y explicar qué pasa en el ordenador representando en la pizarra o la pantalla del ordenador y la memoria (ver Figura 4), de izquierda a derecha: las instrucciones del programa, la pantalla del PC y la memoria en cada paso secuencial del programa.

Para aplicar los guiones, dependiendo de los recursos disponibles, el profesor puede dibujar la tabla en la pizarra paso a paso y explicar cada fila como un paso secuencial en la pizarra; o también podría utilizar un programa como PrimaryCode [31], que ha sido desarrollado específicamente para mostrarlo interactivamente (en ambos casos se introducen los conceptos de entrada/salida, ver Figura 5 para un ejemplo de pantalla de PrimaryCode).

Por ejemplo, como puede verse en la Figura 4, la instrucción 1 (I1) produce la creación de una "caja" en la memoria llamada NombreVariable vacía. La instrucción 2 (I2) envía un mensaje al usuario pidiendo una entrada, el estado de la cajita variable permanece igual. La instrucción

| Programa | Pantalla | Memoria |
|---|---|---|
| **I1: crea nombreVariable** |  | **nombreVariable**  |
| **I2: escribe_en_pantalla ("cuál es tu nombre?")** |  | **nombreVariable**  |
| **I3: guarda (nombreVariable)** |  | **nombreVariable**  |
| **I4: escribe_en_pantalla ("Hello" nombreVariable)** |  | **VariableName**  |

Fig. 4. Ejecución secuencial de las instrucciones (izda.), salida en la pantalla (centro) y estado de la cajita en memoria almacenando el dato (dcha.)


Fig. 5. Ejemplo de pantalla de PrimaryCode [31], aplicación para enseñar Programación desarrollada basándose en las metáforas propuestas

3 (I3) almacena lo que el usuario escribe usando el teclado que se refleja en la pantalla, es este caso su nombre "Mary" se almacena en la variable. Finalmente, la instrucción 4 (I4) produce una frase escrita en la pantalla con un mensaje escrito por el programador "Hello" y el contenido de la variable en memoria (Mary), que produce el mensaje "Hello Mary" en la pantalla.

*A. Guion para introducir a los niños a la programación de instrucciones condicionales.*

(P): "Ahora vamos a aprender como el ordenador toma decisiones: vamos a imaginar que tenemos dos números, ¿os acordáis donde los almacena el ordenador?"

(S): "¡Sí! En variables, en cajitas en la memoria del ordenador."

(P): "¿Pensáis que el ordenador puede saber que cajita tiene el valor mayor?

(S): Contestarán "sí" o "no", o intentarán adivinarlo.

(P): "Vamos a ver si el ordenador es capaz de identificar el valor mayor. Si te preguntaran (Figura 6 arriba) cuál de las dos variables (izquierda) tiene el valor mayor, sabrías que contestar (derecha). Igual lo hace el ordenador, pero debes preguntárselo de una forma un poquito diferente: debes crear una instrucción condicional para que el ordenador decida si el contenido de una cajita es mayor que el de la otra y escribes una instrucción para cuando la condición se cumpla, y otra para cuando no se cumpla (esta rama es opcional).

(P) Continua: "Vamos a ver otro ejemplo de como el ordenador resuelve condicionales. Si te dan las notas, si la nota es mayor que 5 has aprobado y si es menor de 5 has suspendido; el ordenador entiende ese tipo de instrucciones y puede tomar decisiones también. Vamos a ver los ejemplos de la Figura 7. La primera tiene la cajita *NotaV* con el valor 6, en la ejecución (en gris) la parte del programa que se está ejecutando, puedes ver que el PC comprueba si *Nota V* es mayor o igual a 5, en ese caso ejecuta la rama "then" y la salida en la pantalla (dcha.) "Passed"; el resto del código correspondiente a la rama "else" no se ejecuta, porque el ordenador ha tomado ya la decisión de que rama ejecutar. En el siguiente ejemplo, *Nota V* tiene el valor 3, el código que el PC va a ejecutar está también en gris, el PC hace la comprobación con el valor guardado en cajita de memoria *Nota V*, en este caso decide que el valor es menor por lo que ejecuta la rama del "else", la salida por pantalla es "Failed".

| Memoria | Tu (arriba) ordenador (abajo) | Respuestas |
|---|---|---|
| a 7 b 5 | ¿cuál es mayor? | a |
| a 7 b 5 | if a es mayor que b entonces escribe_en_pantalla (a) sino escribe_en_pantalla (b) |   |

Fig. 6. Condicionales, como las resuelves (1ª fila) y el odenador (2ª fila)

A continuación, se pueden hacer más ejemplos de condicionales en los que explicar que el ordenador también puede tomar decisiones más complejas o anidadas. El siguiente ejemplo es sobre esto.

(P): "En Navidad le pedimos a los Reyes Magos o Santa Claus muchas cosas, pero sabemos que recibimos más o menos regalos dependiendo de varios factores. Por ejemplo, vamos a ver…"

(S): "Si me porto bien,..., me lo como todo,…, soy amable"

(P): "de acuerdo, me decís que si os portáis bien, os lo coméis todo y sois amables con los demás recibís muchos regalos, si alguna de esas tareas no se cumplen, recibís menos. Esa clase de decisiones complejas o anidadas también las puede tomar el PC, mirad el ejemplo (Figura 8)".

| Memoria | Instrucc. Programa | Salida PC |
|---|---|---|
| NotaV 6 | if NotaV >= 5 then escribe_en_pantalla ("Passed") else escribe_en_pantalla ("Failed") |   |
| NotaV 3 | if NotaV >= 5 then escribe_en_pantalla ("Passed") else escribe_en_pantalla ("Failed") |   |

Fig. 7. PC ejecutando condicionales. El estado de las variables guardado en memoria (izda.). Las instrucciones que se ejecutan del programa (centro en gris). Salida por pantalla (derecha).

| Memoria | Instrucciones Programa | Salida PC |
|---|---|---|
| portoBien 6<br>comoTodo 7<br>soyMajo 8 | if portoBien >= 5 then<br>if comoTodo >= 5 then<br>if soyMajo >= 5 then<br>escribe_pantalla(`"Santa brings a lot!")<br>else<br>escribe_pantalla ("Santa brings few") |   |
| portoBien 3<br>comoTodo 7<br>soyMajo 5 | if portobien >= 5 then<br>if comoTodo >= 5 then<br>if soyMajo >= 5 then<br>escribe_por_pantalla ("Santa brings a lot!")<br>else<br>escribe_por_pantalla ("Santa brings few") |   |

Fig. 8. PC ejecutando condicionales complejas. Estado de las variables (izda.), instrucciones que se ejecutan (centro) y salida por pantalla (dcha.)

El profesor puede dibujar en la pantalla las 3 partes representadas en Figura 8. En la memoria escribirá las cajitas que representan los valores de las variables (dependiendo de las respuestas de los niños) las instrucciones de programación incluirán estas variables como se muestran en el centro de la figura, y se irán marcando con color las instrucciones que se van ejecutando dependiendo del valor de las variables. En la parte derecha, se muestra la salida por pantalla.

### *B. Guion para introducir a los niños a la programación de Bucles.*

Para introducir el **concepto de bucle** vamos a usar la metáfora de poner la mesa para el número de personas de la familia. El profesor dará varios ejemplos:

(P): "¿Cuántos sois en tu familia?"

(S): "¡cuatro!...¡tres!"

(P): "OK! numPersonas = 4, o numPersonas = 3, entonces después de poner el mantel tendré que repetir las instrucciones 4 ó 3 veces, dependiendo del número de personas en mi familia"

La Figura 9 muestra los dibujos que el profesor tendrá que hacer en la pizarra para introducir el concepto de bucle, (izquierda arriba) las instrucciones para poner la mesa para una persona (izquierda abajo) variable contenedora del número de personas de la familia; (derecha) pictogramas de las veces que repites las instrucciones:

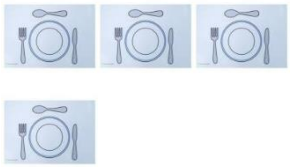
| **Instrucciones** | **Veces que repites las instrucciones** |
|---|---|
| **pon el plato, tenedor, cuchillo y cuchara 4 veces (nPersonas)** |  |
| **Pon el plato, tenedor, cuchillo y cuchara 3 veces (nPersonas)** |  |

Fig. 9. Representa como el niño repite un conjunto de instrucciones varias veces para poner la mesa para los miembros de su familia

| **Memoria** | | **Instrucciones de Programa** | **Salida Pantalla** |
|---|---|---|---|
| **Antes** | **Después** | | |
| **ejecución** | | | |
| **variable** **0** | **variable** **1** | **Mientras (variable <= 3) hacer Escribe_pantalla ("Hello") Suma 1 a variable finMientras** |   |
| **variable** **1** | **variable** **2** | **Mientras (variable <= 3) hacer Escribe_pantalla ("Hello") Suma 1 a variable finMientras** |   |
| **variable** **2** | **variable** **3** | **Mientras (variable <= 3) do Escribe_pantalla ("Hello") Suma 1 a variable finMientras** |   |

Fig. 10. Metáforas para el concepto de bucle (grupo de metáforas M4)

Después de comprender el concepto de bucle con la metáfora anterior, el profesor deberá mostrar ejemplos de cómo el PC repite un conjunto de instrucciones en la pizarra o la pantalla, como anteriormente (Figura 10), el estado de las variables en memoria (izquierda) ahora introducimos el concepto de **antes de la ejecución y después de la ejecución**; las instrucciones del programa que se están ejecutando (centro) y la salida en la pantalla (derecha).

## IV. CASO DE ESTUDIO PILOTO: EVALUACIÓN

62 estudiantes (41,9% niños y 58,1% niñas) agrupados en 4º (23 niños, 37%), 5º (17 niños, 27,4%) y 6º (22 niños, 35,6%) curso de Educación Primaria (de 9 a 11 años), como se muestra en la Figura 11, participaron en una encuesta para analizar su opinión respecto al uso de metáforas para enseñarles conceptos básicos de programación.

Cuando a los niños se les preguntó si querían crear un programa siguiendo los pasos, como los niños que participan en Master Chef y siguen los pasos de las recetas para cocinar, 55 de los 62 niños (88.7%) dijeron que sí (100% de los estudiantes de cuarto, 88.2% de los estudiantes de quinto, y 90.9% de los estudiantes de sexto).

Cuando a los niños se les preguntó por la **primera metáfora del grupo M1**, esto es, su opinión sobre comparar programar como seguir una receta, **63.3% respondió que la metáfora le facilitaba comprender el concepto**. 1 estudiante de los 23 matriculados en 4º, 4 de los 17 matriculados en 5º, y 1 de los 22 matriculados en 6º manifestaron que la metáfora les resultaba compleja. 2 niños de 6º resaltaron que les sorprendía esta metáfora, la encontraban divertida y nunca se les hubiese ocurrido la relación entre programar y cocinar. 2 niños de los 23 matriculados en 4º, 0 de los 17 matriculados en 5º y 2 de los 22 matriculados en 6º dijeron que ellos hubiesen preferido otra explicación.

Cuando a los niños se les preguntó por la **segunda metáfora del grupo M1**, esto es, su opinión sobre comparar la memoria del ordenador con una despensa, el **65.8% respondió que la metáfora le facilitaba comprender el concepto**. 1 de los 23 estudiantes matriculados en 4º, 2 de los 17 matriculados en 5º, y 1 de los 22 matriculados en 6º dijeron que les parecía difícil entender la metáfora. 2 niños en 4º, 1 en 5º, y 3 en 6º hubiesen preferido otra explicación.

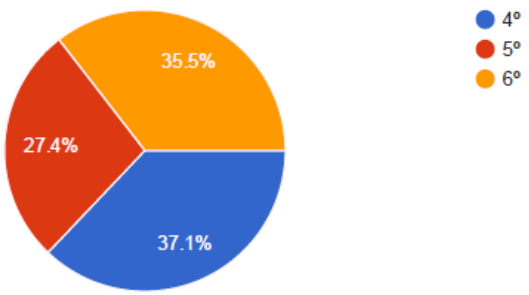



Fig. 11. Distribución de los 62 estudiantes en cursos

Cuando a los niños se les preguntó por la **tercera metáfora del grupo M1**, esto es, su opinión sobre comparar cajitas con variables, **el 73.7% de los estudiantes respondió que la metáfora le facilitaba comprender el concepto**. 6 estudiantes de los 23 matriculados en 4º, 0 de los 17 matriculados en 5º, y 0 de los 22 matriculados en 6º manifestaron que la metáfora les resultaba compleja. 0 niños de 4º, 2 niños de 5º y 2 niños de 6º dijeron que ellos hubiesen preferido otra explicación.

Cuando a los niños se les preguntó por la **cuarta metáfora (grupo M2)**, esto es, su opinión sobre comparar entrada/salida con teclado/pantalla, el **89.9% respondió que la metáfora le facilitaba comprender el concepto**. 2 de los 23 estudiantes matriculados en 4º, 0 de los 22 matriculados en 5º, y 3 de los 22 matriculados en 6º dijeron que les parecía difícil entender la metáfora. 0 niños en 4º, 0 en 5º, y 2 en 6º hubiesen preferido otra explicación.

Cuando a los niños se les preguntó por la **quinta metáfora (grupo M3)**, esto es, su opinión sobre comparar condicionales con tomar decisiones, el **89.6% respondió que la metáfora le facilitaba comprender el concepto**. 1 de los 23 estudiantes matriculados en 4º, 3 de los 17 matriculados en 5º, y 2 de los 22 matriculados en 6º dijeron que les parecía difícil entender la metáfora. Ningún niño indicó que hubiese preferido otra explicación.

Cuando a los niños se les preguntó por la **sexta metáfora (grupo M4)**, esto es, su opinión sobre comparar bucles con repeticiones, el **89.6% respondió que la metáfora le facilitaba comprender el concepto**. 1 de los 23 estudiantes matriculados en 4º, 3 de los 17 matriculados en 5º, y 2 de los 22 matriculados en 6º dijeron que les parecía difícil entender la metáfora. Ningún niño indicó que hubiese preferido otra explicación.

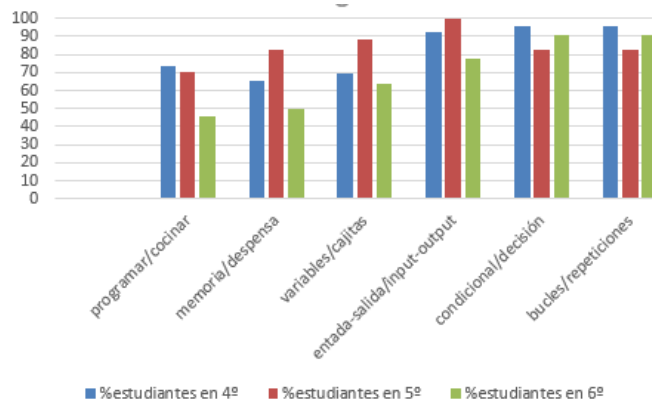



Figura 12. Porcentaje de estudiantes que encontraron útiles las metáforas

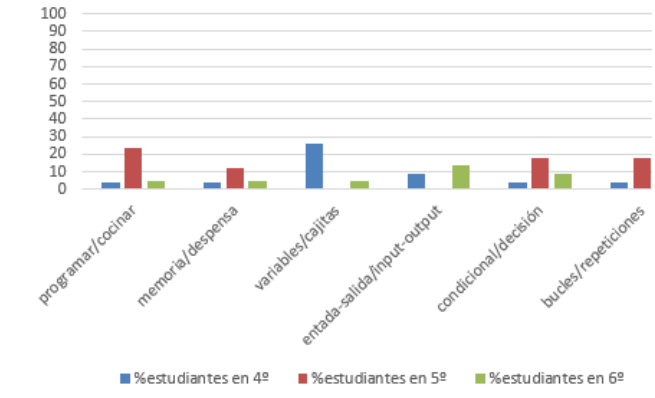



Figura 13. Porcentaje de estudiantes que encontraron difíciles las metáforas

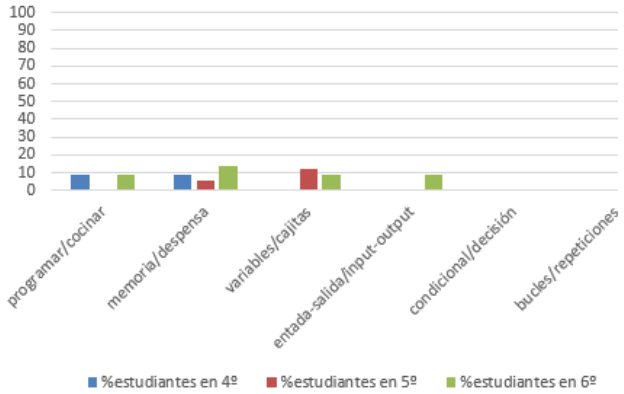



Figura 14. Porcentaje de estudiantes que solicitaron una explicación diferente

explicación.

La Figura 12 muestra un gráfico de barras con la opinión de la utilidad de los estudiantes matriculados en cada curso. Como se puede apreciar, en general, los estudiantes encontraron útiles las metáforas.

La Figura 13 muestra el porcentaje de estudiantes que encontraron difícil comprender las metáforas. Como se puede apreciar, menos del 30% de los estudiantes encontraron difícil estas metáforas, incluso aquellos que estaban matriculados en 4º. Por último, la Figura 14 muestra el porcentaje de estudiantes que solicitaron una forma alternativa de explicación. Como se puede apreciar, menos del 10% de los estudiantes necesitaron otra explicación

## V. CONCLUSIONES

Hay un interés mundial en investigar cómo enseñar conceptos básicos de programación a los niños, y desarrollar su pensamiento computacional. Sin embargo, no se ha encontrado todavía ninguna metodología que pueda considerarse cómo la más adecuada para guiar a los profesores en esta tarea. Según la revisión realizada, normalmente los profesores usan Scratch o ejercicios de Code.org, pero no tienen una guía de como transmitir los conceptos a más alto nivel.

En este artículo, el objetivo es proporcionar esta guía a los profesores y a los investigadores basada en el uso de metáforas. Según el RD 126/2014, los estudiantes a partir de 4º de Primaria entienden el concepto de metáfora, y según la revisión de literatura realizada, las metáforas han sido útiles en la enseñanza de programación en niveles universitarios.

La propuesta ha sido validada por 62 niños de 4º, 5º y 6º de Educación Primaria entre 9 y 11 años de edad, que las encontraron útiles (en más de un 65% de los casos), fueron capaces de comprender las metáforas (menos de un 30% de los estudiantes las encontraron difíciles), y sólo en menos del 10% de los casos pidieron una explicación alternativa.

Además, se prueba la validez a nivel de eficacia educativa de la metáfora puesto que los estudiantes aprendieron, como se ha podido cuantificar en los resultados de un pre-post test que se les pasó antes y después de seguir la metodología propuesta tanto sin pizarra como con PrimaryCode [31] (t=-3.668, p=0.001 y t=-3.205, p=0.003 respectivamente).

El profesor también validó la metodología seguida. En una entrevista que se le realizó, a la pregunta de que le parecía la enseñanza de la programación basada en metáforas, nos respondió textualmente: “Respecto a lo de la

explicación yo creo que sí está bien porque además se trabajan los textos instructivos y en concreto el de las recetas". Se pretende seguir aplicando la metodología en más colegios y trabajando conceptos informáticos desde edades tempranas para poder ver su impacto en el desarrollo del pensamiento computacional.

## AGRADECIMIENTOS

Queremos expresar nuestra gratitud al colegio y a los proyectos TIN2015-66731-C2-1-R y S2013/ICE-2715.

## REFERENCIAS

[1] F. J. García-Peñalvo, "Proyecto TACCLE3 – Coding," in XVIII Simposio Internacional de Informática Educativa, SIIE 2016, F. J. García-Peñalvo and J. A. Mendes, Eds. no. 222) Salamanca, España: Ediciones Universidad de Salamanca, 2016, pp. 187-189.

[2] F. J. García-Peñalvo, "A brief introduction to TACCLE 3 – Coding European Project," in 2016 International Symposium on Computers in Education (SIIE 16), F. J. García-Peñalvo and J. A. Mendes, Eds. USA: IEEE, 2016.

[3] F. J. García-Peñalvo, D. Reimann, M. Tuul, A. Rees, and I. Jormanainen, "An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers," TACCLE3 Consortium, Belgium. 2016. doi:10.5281/zenodo.165123

[4] P.Y. Chao, Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. Computers & Education, 95, 2016, pp. 202-215.

[5] C. Brackmann, D. Barone, A. Casali, R. Boucinha, S. Muñoz-Hernandez, "Computational thinking: Panorama of the Americas", In International Symposium on Computers in Education (SIIE), 2016, pp. 1-6.

[6] S. Campe, J. Denner, Programming Games for Learning: A Research Synthesis. Paper presented at the annual meeting of the American Educational Research Association, Chicago, 2015.

[7] I. Ouahbi, F. Kaddari, H. Darhmaoui, A. Elachqar, S. Lahmine, "Learning basic programming concepts by creating games with scratch programming environment", Procedia-Social and Behavioral Sciences, 191, 2015, pp. 1479-1482.

[8] M. Jovanov, E. Stankov, M. Mihova, S. Ristov, M. Gusev, "Computing as a new compulsory subject in the Macedonian primary schools curriculum", In Global Engineering Education Conference (EDUCON), 2016, IEEE, pp. 680-685.

[9] E. Lahtinen, K. Ala-Mutka, H.M. Järvinen, "A study of the difficulties of novice programmers", in ACM SIGCSE Bulletin, vol. 37, no. 3, 2005, pp. 14-18.

[10] D. Ginat, "On novice loop boundaries and range conceptions", Computer Science Education, 14(3), 2004, pp. 165-181.

[11] O. Seppälä, L. Malmi, A. Korhonen, "Observations on student misconceptions—A case study of the Build–Heap Algorithm", Computer Science Education, 16(3), 2006, pp. 241-255.

[12] L.J. Barker, C. McDowell, K. Kalahar, "Exploring factors that influence computer science introductory course students to persist in the major", In ACM SIGCSE Bulletin, Vol. 41, No. 1, 2009, pp. 153-157.

[13] N.J. Coull, I.M. Duncan, "Emergent requirements for supporting introductory programming", Innovation in Teaching and Learning in Information and Computer Sciences, 10(1), 2011, pp. 78-85.

[14] A. Yadav, S. Gretter, S. Hambrusch, P. Sands, "Expanding computer science education in schools: understanding teacher experiences and challenges", Computer Science Education, 2016, pp. 1-20.

[15] F. Heintz, L. Mannila, T. Färnqvist, "A review of models for introducing computational thinking, computer science and computing in K-12 education", IEEE Frontiers in Education Conference, 2016, pp. 1-9.

[16] J.P. Sanford, A. Tietz, S. Farooq, S. Guyer, R.B. Shapiro, "Metaphors we teach by", In Proceedings of the 45th ACM technical symposium on Computer science education, 2014, pp. 585-590.

[17] INTEF, Marco Común de Competencia Digital Docente. Spanish Ministry of Education, Culture and Sports. Available on-line at: https://goo.gl/7pvLve (última visita: 30 de marzo, 2017).

[18] M. Resnick, J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, et al., Scratch: Programming for all. Communications of the ACM, 52(11), 2009, pp. 60-67.

[19] K. Brennan, M. Resnick, "New frameworks for studying and assessing the development of computational thinking". In Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada, 2012, pp. 1-25.

[20] A. Sović, T. Jagušt, D. Seršić, "How to teach basic university-level programming concepts to first graders?", In Integrated STEM Education Conference (ISEC), 2014, IEEE, pp. 1-6.

[21] F. Kalelioğlu, "A new way of teaching programming skills to children: Code. Org", Computers in Human Behavior, 52, 2015, pp. 200-210.

[22] G. Lakoff, M. Johnson, "Metaphors we live by". University of Chicago press, 2008.

[23] N.A. Paris, S.M. Glynn, "Elaborate analogies in science text: Tools for enhancing preservice teachers' knowledge and attitudes", Contemporary Educational Psychology, 29(3), 2004, pp. 230-247.

[24] G.P. Thomas, C.J. McRobbie, "Using a metaphor for learning to improve students' metacognition in the chemistry classroom", Journal of Research in Science Teaching, 38(2), 2001, pp. 222-259.

[25] P. Boero, L. Bazzini, R. Garuti, "Metaphors in teaching and learning mathematics: a case study concerning inequalities". In Pme Conference, Vol. 2, 2001, pp. 2-185.

[26] R.T. Putnam, D. Sleeman, J.A. Baxter, L.K. Kuspa, "A summary of misconceptions of high school Basic programmers", Journal of Educational Computing Research, 2(4), 1986, pp. 459-472.

[27] R. Jiménez-Peris, C. Pareja-Flores, M. Patiño-Martínez, J.A. Velázquez-Iturbide, "The locker metaphor to teach dynamic memory", In ACM SIGCSE Bulletin, Vol. 29, No. 1, 1997, pp. 169-173.

[28] W.W. Milner, "A broken metaphor in Java", ACM SIGCSE Bulletin, 41(4), 2010, pp. 76-77.

[29] D.H. Clements, B.K. Nastasi, S. Swaminathan, "Young children and computers: crossroads and directions from research". Young children, 48(2), 1993, pp. 56–64.

[30] RD 126/2014. Basic Curriculum Primary Education in Spain. Available on-line at: https://goo.gl/nHRs8n (última visita: 6 de marzo, 2018).

[31] PrimaryCode. http://www.lite.etsii.urjc.es/tools/primarycode/ (última visita: 6 de marzo, 2018).

**Diana Pérez Marín** es Doctora en Ingeniería Informática y Telecomunicación por la Universidad Autónoma de Madrid desde el año 2007. Actualmente, es Profesor Contratado Doctor del Departamento de la Escuela Técnica Superior de Ingeniería Informática de la Universidad Rey Juan Carlos. Su principal línea de interés en investigación es el uso de los ordenadores para la educación con 24 artículos indexados (Thomson Reuters Researcher ID L-4100-2014), H-index 8, y 141 citas (Scopus: 14042379400).

**Raquel Hijón Neira** tiene el Doctorado Europeo en Informática por la Universidad Rey Juan Carlos, Madrid, España. Ha trabajado como Ingeniera Informática durante cinco años, y enseñando en la Universidad durante 17 años. Actualmente, es Profesor Contratado Doctor en la Universidad Rey Juan Carlos, y miembro del Laboratorio para la Investigación en Tecnologías Educativas (LITE). Sus áreas de investigación son la innovación en la enseñanza de la programación, Interacción Persona-Ordenador y juegos serios. Dr. Hijón-Neira obtuvo el galardón a la mejor tesis otorgado por la IEEE.

**Mercedes Martín-Lope** es Profesor de Metodología de la Investigación Educativa, Didáctica de las Matemáticas y Diseño Curricular. Es Doctora en Métodos de Investigación y Diagnóstico Educativo por la Universidad Rey Juan Carlos (2015). Martín-Lope tiene una larga experiencia en enseñanza pre-universitaria (1998-2005). A partir del año 2003 comenzó a dar clases en la Universidad. Actualmente, es Coordinadora del Grado de Infantil de la Universidad Rey Juan Carlos. Dr. Martín-Lope ha participado en varios congresos internacionales y coordinado proyectos de Innovación Educativa.