RMIE, 2015, VOL. 20, NÚM. 65, PP. 607-641 (ISSN: 14056666)

Investigación

ESTADO DEL ARTE SOBRE EXPERIENCIAS DE ENSEÑANZA DE PROGRAMACIÓN A NIÑOS Y JÓVENES PARA EL MEJORAMIENTO DE LAS COMPETENCIAS MATEMÁTICAS EN PRIMARIA

CARLOS ANDRÉS PALMA SUÁREZ / ROMÁN EDUARDO SARMIENTO PORRAS

Resumen:
Este artículo presenta una revisión del estado del arte sobre experiencias de enseñanza de programación dirigidas a niños y jóvenes para mejorar sus habilidades matemáticas. La investigación se realizó con el fin de proponer un modelo para enseñar la elaboración de macroinstrucciones a partir del entendimiento de procesos lógico-matemáticos, y mejorar la resolución de problemas en estudiantes de 5° grado de primaria, particularmente, sobre la comprensión y el desarrollo de sus primeros algoritmos. En la indagación se encontraron importantes aspectos para considerar en el modelo que pretendemos proponer: temáticas, contextos, herramientas y técnicas adecuados para la enseñanza de programación a niños, así como algunos ejemplos de modelos de evaluación de resultados de este tipo de experiencias.

Abstract:
This article presents a review of the state of the art of experiences in teaching programming to children to improve their math skills. The research was carried out to propose a model for teaching children to elaborate macro-instructions based on their understanding of logical mathematical progresses, and to improve problem solving among fifth graders, especially involving the initial understanding and development of algorithms. The study found important aspects to consider in the model that we hope to propose: adequate topics, contexts, tools, and techniques for teaching programming to children, as well as some examples of models to evaluate the results of this type of experiences.

Palabras clave: programación, niños, algoritmos, enseñanza, razonamiento matemático, Colombia.
Keywords: programming, children, algorithms, teaching, mathematical reasoning, Colombia.

Carlos Andrés Palma Suárez es estudiante de maestría y Román Eduardo Sarmiento Porras es docente-investigador, ambos en el Grupo de Investigación de Preservación e Intercambio Digital de Información y Conocimiento - PRISMA de la Universidad Autónoma de Bucaramanga. Avenida 42, núm. 48-11, Bucaramanga, Colombia. CE: cpalma@unab.edu.co / rsarmiento@unab.edu.co

Introducción

Durante los últimos años se ha visto un deterioro notable en la calidad educativa en el área de matemáticas, en Colombia, así lo demuestran los resultados de las pruebas SABER de los años 2002, 2005 y 2009 en el nivel quinto de educación básica primaria (Instituto Colombiano para la Evaluación de la Educación, ICFES, 2010). Revisando los niveles de desempeño en dicha área en el departamento de Santander y presentando una comparación con los resultados generales del país, se observa claramente que no se logra alcanzar un nivel al menos satisfactorio en más de 50% de la población estudiantil, ni en el contexto departamental ni nacional.

Una manera para contribuir a la solución de este grave problema no es encaminar únicamente el uso de la tecnología para apoyar los procesos académicos, sino que también se ha buscado enseñarla, por lo que desde hace mucho tiempo se ha hecho hincapié en ello, particularmente en la enseñanza de la programación en los niños.

Uno de los aspectos que abarca la enseñanza tecnológica es la comprensión de los procesos lógicos como fases secuenciales de operaciones que transforman recursos, el fin es cumplir con ciertos objetivos y lograr un resultado esperado, necesitando para ello tomar decisiones que asocian "propósitos, recursos y procedimientos para la obtención de un producto o servicio" (Ministerio de Educación Nacional, 2008). Esta definición concuerda en buena parte con uno de los cinco procesos generales de la actividad lógica-matemática, según los Estándares Básicos de Competencias en Matemáticas (Ministerio de Educación Nacional, 2006). De acuerdo con ellos, la formulación, comparación y ejercitación de procedimientos es un proceso en el que se busca enseñar a los estudiantes a construir y ejecutar en forma precisa un conjunto de procedimientos secuenciales y lógicos llamados algoritmos.

Enseñanza de la programación para el mejoramiento de las competencias matemáticas

Los cursos de fundamentos o de introducción de programación de computadores impartidos en colegios o universidades se han basado siempre en la enseñanza de un lenguaje ya sea de programación técnico o sustentado sobre un pseudocódigo (Joyanes Aguilar, 2002). Este último permite utilizar una serie de instrucciones que representan sencillos procesos a ser ejecutados en forma secuencial o un conjunto de procesos más complejos

(o macroinstrucción). Existen herramientas que permiten trabajar con esta metodología, diseñadas para asistir a los estudiantes en sus primeros pasos en programación mediante pseudolenguajes creados incluso en los idiomas locales y/o complementados con editores gráficos para generación de código. Sin embargo, el objetivo particular de la enseñanza de la programación no radica solo en ilustrar cómo se escriben secuencias de instrucciones con una sintaxis particular para que el computador las ejecute, sino que se busca centrar la atención en los conceptos fundamentales de lo que se conoce como pensamiento computacional o algorítmico, que permite que los estudiantes establezcan una serie de pasos para resolver un problema (Grover, 2009).

Cientos de estrategias para resolución de problemas en diversas ramas del saber, especialmente en matemáticas, se basan en la aplicación de pasos para cumplir un objetivo (Woods, 2000). Abelson, Sussman y Sussman (1996) abordan la importancia de la enseñanza de la programación de computadores, a tal punto de presentarla como un complemento de la matemática, dado que mientras "la matemática provee un marco de trabajo para tratar en forma precisa con nociones de tipo -qué-, la computación provee un marco de trabajo para tratar en forma precisa con nociones de tipo -cómo". Iniciativas en otros países como los proyectos Code.org (2013), o Scratch entre muchos otros, buscan que mediante la programación, los estudiantes aprendan a pensar creativamente, razonar sistemáticamente y trabajar colaborativamente, incluso desde temprana edad (MIT Media Lab, 2013). En palabras de muchos investigadores, la enseñanza de la programación puede hacer que los niños mejoren las habilidades de pensamiento y resolución de problemas. De acuerdo con Delgado *et al*. (2013): "la programación de ordenadores aporta algo positivo y diferente a la formación de una persona: hábitos y conocimientos que tienen un cierto valor práctico en el día a día de, como mínimo, cualquier persona que viva en un entorno urbano del primer mundo".

Se podría afirmar entonces que la enseñanza de la formulación de procesos secuenciales en los niños atañe no solo al área de conocimiento tecnológico sino también a la de matemática. Una propuesta de iniciar en los niños de quinto grado de primaria la enseñanza de elaboración de instrucciones secuenciales no suena ahora tan descabellada sino que se vuelve una herramienta que ayudaría a generar y mejorar competencias en las áreas de matemáticas y de tecnología. Surge de aquí este gran interrogante: ¿Qué características deben tenerse en cuenta para generar un marco de trabajo

en el que se enseñe el desarrollo de macroinstrucciones con el propósito de mejorar el desarrollo de las capacidades de razonamiento para resolución de problemas en los niños de grado quinto de educación básica primaria?

Con base en esa pregunta de investigación, se realizó una revisión de la literatura en la cual se encontraron documentos y experiencias relevantes con la enseñanza de lenguajes de programación educativos (buscando las palabras clave: *teaching programming kids*). Mediante las bases de datos Scopus, ACM Digital Library, ScienceDirect, IEEExplore y el motor de búsqueda Google Scholar, se encontró un grupo de publicaciones entre artículos y sitios web que, a pesar de utilizar las palabras clave, muchas pertenecían a otras áreas de investigación, como psicología o medicina, o a otros niveles académicos que presentaban una complejidad temática más avanzada. Se seleccionaron las adecuadas tomando como prioridades de inclusión la temática relacionada con ciencias de la computación y enseñanza de programación para niños y/o jóvenes, incluyendo experiencias aplicadas en estudiantes de primer año de universidad. Se le dio mayor relevancia a los documentos más citados y a publicaciones más recientes con una ventana de observación de un lapso de diez años. Esta revisión de literatura se abordó bajo los siguientes temas:

- Experiencias de enseñanza de programación para niños y jóvenes.
- Temáticas y contextos adecuados de enseñanza de la programación para niños.
- Caracterización de herramientas adecuadas para la programación en niños.
- Técnicas de aplicación de la enseñanza de la programación en el aula.
- Modelos de evaluación de resultados en experiencias generadas de enseñanza.

Experiencias de enseñanza de programación para niños y jóvenes

La revisión de la literatura evidenció varias experiencias de enseñanza de programación en niños y jóvenes en diferentes instituciones alrededor del mundo. Se generó una caracterización inicial a partir de dichas experiencias, que comprendió los siguientes elementos:

- Autor
- Año de la experiencia
- País o países de aplicación

- Herramienta seleccionada
- Temática presentada y contexto en el que aplica
- Población objetivo (estudiantes) y presaberes
- Técnica aplicada
- Modelo pedagógico aplicado

Se elaboró un cuadro comparativo (ver cuadro 1) que permitió identificar diferentes características de 18 experiencias de enseñanza, y con lo cual se obtuvo un panorama más concreto sobre los temas abordados en este informe. En la revisión de esta literatura se evidenció que en muchos países europeos (por ejemplo, Austria, República Checa, Israel, Serbia, Dinamarca, Eslovaquia, Rusia), así como en Norteamérica (Estados Unidos, Canadá) y en Asia (Taiwán) existe la preocupación por enseñar programación de computadores desde temprana edad, pero no se presentó un consenso en el que se estableciera un currículo estándar generado por sus gobiernos sobre las temáticas a tratar o los niveles académicos desde donde se debe comenzar dicha enseñanza.

Estas experiencias abarcaron todos los niveles educativos, desde preescolar hasta los primeros de educación superior, implicando su aplicación en contextos temáticos diferentes de acuerdo con su nivel. Ninguna temática ha sido aplicada de manera forzada sino que se adaptaron como tratados en el ámbito de cada institución y país. Si se compara con el contexto curricular colombiano, se encuentra que, por ejemplo, en educación primaria no se enseña de manera obligatoria el concepto del funcionamiento de circuitos eléctricos. Es necesario hacer una revisión de las herramientas y las metodologías más adecuadas para la enseñanza de macroinstrucciones teniendo en cuenta que el contexto aplicado debe estar acorde con el contenido presentado en los lineamientos curriculares nacionales en el área de matemáticas. Se encontraron además dos experiencias (Alt, Astrachan, Forbes, Lucic y Rodger, 2006; Lin, Zhang, Beck y Olsen, 2009) que presentaron diferentes alternativas para el uso y motivación hacia las ciencias de la computación para la resolución de problemas reales incluso sin necesidad de enseñar programación. Alt *et al*. (2006) utilizaron para ello el análisis de datos estadísticos con el objeto de conocer el comportamiento de las redes sociales, mientras que Lin *et al*. (2009) emplearon el contexto de la bioinformática y el análisis genético de cadenas de código ADN para la comprensión de relaciones evolutivas.

CUADRO 1

Comparativo de experiencias de la enseñanza de la programación a niños y jóvenes

Autores y año	**País o países de aplicación**	**Herramienta seleccionada**	**Temática presentada y contexto en el que aplica**	**Población objetivo (estudiantes) y presaberes**	**Técnica aplicada**	**Modelo pedagógico aplicado**
Abramovich, 2013	Austria, Canadá, Rep. Checa, Israel, Serbia, EUA	Utilitarios de matemáticas: GeoGebra, VisualMath, Maple	Matemáticas; contexto dependiente del tema	Diversos niveles educativos; con conocimientos previos de matemáticas	Resolución de problemas según el método Polya	Ninguno propuesto
Liu, Cheng y Huang, 2011	Taiwán	Juego de simulación Train B&P	Programación; contexto sobre control de transporte de un sistema de ferrocarril	Primer año de universidad; sin experiencia en programación	Diseño experimental, simulación y generación de hipótesis, interpretación de datos, resolución de problemas según el método Polya	Prueba y error, aprendizaje por ejemplo, razonamiento, aprendizaje por construcción
Valente, 2004	Dinamarca	Cartas computacionales	Teoría de las ciencias de la computación y la información; contexto sobre circuitos básicos y dinámicos, problemas de probabilidad y transmisión de información	Primaria, de 8 a 10 años; sin experiencia en programación, incluso con limitación de formación matemática	Armado de circuitos de flujo (ubicación de cartas) y aplicación de reglas formales, armado de circuitos especiales para reutilización	Manipulación directa y aprendizaje por realización (construcción), desarrollo de proyectos top-down y bottom-up
Felleisen, Findler, Flatt, y Krishnamurthi, 2009	EUA	Entorno de desarrollo DrScheme	Matemática plana, álgebra y programación funcional; contextos diferentes para crear simulaciones, animaciones y juegos	Bachillerato, de 10 a 14 años; con presaberes de matemática básica	Programación funcional mediante modelos matemáticos	Modelamiento matemático
Wolz, Leitner, Malan, y Maloney, 2009	EUA	Entorno de desarrollo Scratch	Programación básica y avanzada (eventos, concurrencia, hilos); contexto diferente de acuerdo con la temática	Primer semestre universidad, con conocimiento nulo o muy básico en programación	Enseñanza de programación mediante armado, transición a lenguaje de programación real	Aprendizaje por construcción

CUADRO 1 / CONTINUACIÓN

Tomcsányiová y Tomcsányi, 2011	Eslovaquia	Cada tarea bajo un applet	Uso de TIC para comunicación, diagramas de flujo, resolución de problemas, pensamiento algorítmico, principios de TIC, sociedad de la información; varios contextos con diferentes niveles de dificultad	Primaria, estudiantes de 2° a 4° (7-9 años), con habilidad mental	Desarrollo de torneo competitivo sobre resolución de tareas interesantes y evaluación de resultados	Competencia y evaluación. Trabajo con objetos concretos sin uso de abstracción. Clasificación, adecuación, Formulación de tareas
Doerschuk, Liu y Mann, 2012	EUA	Plataforma GreenFoot	Conceptos básicos de programación, POO; contexto para programación de juegos	Jóvenes de grados 10° y 11°; ningún conocimiento sobre programación	Campamento de programación intensivo	Enseñanza de material instruccional preparado por estudiantes de niveles superiores
Rogozhkina y Kushnirenko, 2011	Rusia	Entorno PiktoMir	Conceptos fundamentales de programación; contexto sobre juego de control de robot para llenado de casillas	Educ. infantil (5.5 a 7 años); manejo de mouse, ninguna o poca habilidad de lectura	Juego para cumplimiento de objetivos, sin uso y con uso de computador	Instrucción básica, pruebas escritas gráficas y aplicadas en computador
Burke y Kafai, 2010	EUA	Entorno de desarrollo Scratch	Programación básica, orientación a objetos. Contexto: creación de historias	Estudiantes de educación media, entre 10 y 14 años	Programación básica para narración de historias y creación de juegos	Generación de sentido narrativo secuencial, capacidad de escritura
Javidi y Sheybani, 2009	EUA	Entornos de programación Kahootz y Squeak	Programación para videojuegos. Contexto: temas sobre conservación del medio ambiente y salud	Estudiantes de educación media de áreas urbanas y rurales	Diseño de videojuegos, creación de ideas con historias, preparación para Liga Lego	Trabajo en equipo, exploración de software instruccional, autorreflexión
Kelleher, Pausch, y Kiesler, 2007	EUA	Entorno de desarrollo Storytelling Alice	Programación básica. Contexto: creación de historias	Estudiantes mujeres de educación básica y media (5° a 9°)	Enseñanza de herramientas y conceptos de programación, quiz de conceptos y actitudes, construcción de historias	Enseñanza a través de tutorial, uso de constructos de programación básica
Lee y Ko, 2011	EUA	Juego de programación Gidget	Programación básica, sintaxis para programación. Contexto: juego para ayudar a un robot a corregir sus fallas de programación para cumplir misiones	Personas desde bachillerato hasta doctorado alrededor del mundo, sin conocimiento previo de programación	Enseñanza de comandos básicos, composición de comandos más complejos, comprensión de errores	Enseñanza a través de tutorial, presentación de variaciones sintácticas con errores. Aprendizaje por error

(CONTINÚA)

CUADRO 1 / CONTINUACIÓN

Meyers, Cole, Korth y Pluta, 2009	EUA	Entorno de programación Processing	Programación básica y avanzada (funciones, gráficos y fractales), procesamiento de sonidos. Contexto: análisis y generación de melodías e imágenes	Estudiantes entre 12 y 17 años, con proficiencia en matemáticas y conocimiento en música	Enseñanza de programación y música computacional. Representación de imágenes, manipulación y sintetización de sonidos en forma electrónica	Construcción de imágenes y música
Rizvi *et al*., 2011	EUA	Entorno de programación Scratch	Programación básica. Implementación de proyectos. Contexto: creación de proyecto multimedia con documentación	Estudiantes recién ingresados a universidad	Curso de programación utilizando Scratch, explicación de constructos básicos y avanzados de programación	Aprendizaje tradicional con uso de TIC
Rodger *et al*., 2010	EUA	Entorno de programación Alice	Matemáticas, música, programación. Contexto: uso de videojuegos para diferentes contextos dependiendo de la temática	Todos los niveles educativos, desde primaria hasta universidad	Implementación de proyectos 3D de tipo videojuegos, para aplicar en matemáticas, programación y música	Aprendizaje tradicional con uso de TIC
Sipitakiat y Nusen, 2012	Tailandia	Sistema de construcción por bloques físicos Robo-Blocks	Programación para robótica. Contexto: creación de minirobots	Niños entre 8 y 9 años	Creación de robots mediante construcción de bloques. Enseñanza de modelo de depuración	Aprendizaje por construcción
Tarkan *et al*., 2010	EUA	Lenguaje de programación para niños Toque	Diseño de órdenes secuenciales. Contexto: Escenarios de cocina para preparación de recetas	Niños entre 7 y 11 años	Proceso de diseño e implementación de recetas de cocina	Aprendizaje por construcción
Zuckerman, Arida y Resnick, 2005.	EUA	Bloques de construcción electrónicos FlowBlocks y SystemBlocks	Conceptos matemáticos concretos y abstractos. Contexto: construcción de flujos para diversos escenarios	Niños entre 4 y 11 años	Estructuras genéricas comparadas con objetos reales, nivel de abstracción, asociación semántica, analogías	Manipulativos digitales inspirados en la metodología Montessori

Elaboración propia.

Temáticas y contextos para la enseñanza de la programación para niños

La enseñanza de la programación se ha utilizado en muchos contextos, entre ellos se incluyen creación de historias (Burke y Kafai, 2010; Kelleher, Pausch y Kiesler, 2007), conservación del medio ambiente y salud (Javidi y Sheybani, 2009), procesamiento de sonidos musicales (Meyers, Cole, Korth y Pluta, 2009), escenarios de cocina para preparación de recetas (Tarkan *et al*., 2010), control de transporte (Liu, Cheng y Huang, 2011), los clásicos contextos matemáticos –como modelamiento (Abramovich, 2013; Felleisen, Findler, Flatt y Krishnamurthi, 2009)– y tecnológicos –como programación de robots reales (Sipitakiat y Nusen, 2012) y de computadores– (Wolz, Leitner, Malan y Maloney, 2009), uso de videojuegos para el cumplimiento de objetivos (Lee y Ko, 2011; Rodger *et al*., 2010; Rogozhkina y Kushnirenko, 2011; Tomcsányiová y Tomcsányi, 2011), creación de videojuegos (Doerschuk, Liu y Mann, 2012), flujos (Zuckerman, Arida y Resnick, 2005; Valente, 2004), y multimedia (Rizvi, Humphries, Major, Jones y Lauzun, 2011).

A pesar de los diversos contextos y temáticas centrales, dentro de éstos siempre se enseñó la organización de un conjunto de órdenes secuenciales para la ejecución de tareas particulares. Este tópico particular llamado algoritmia es enseñado a través de un conjunto de temas más específicos. De acuerdo con muchas de las investigaciones encontradas, estos conjuntos de temas son lo que se denomina estructuras de programación. Revisando éstas se realizó una compilación de temas y se organizó de la siguiente forma:

- Tipos de variables (numéricas, textuales, booleanas)
- Sintaxis (lenguaje formal) de la herramienta
- Reusabilidad de ejemplos y experimentación
- Secuencialidad, diseño de órdenes (ideación y formulación de modelos computacionales), analogías con escenarios reales
- Entradas y salidas de información
- Entendimiento de respuestas del sistema
- Depuración (corrección de errores) pre-ejecución, en ejecución, post-ejecución
- Estructuras de control: condicionales, iteraciones, anidaciones
- Arreglos y matrices
- Funciones y expresiones matemáticas

- Subrutinas
- Gráficas y coordenadas cartesianas
- Imágenes, animaciones y sonido (multimedia)
- Interactividad con teclado y mouse, programación de eventos
- Orientación a objetos, clases, herencia
- Transición a un lenguaje de programación (Java, C++)
- Simulación de comunicación (sincronización)
- Actividades de no programación (diseño de escenarios e interfaz)

Determinar la edad adecuada en los niños para explicar estos conceptos es fundamental. Según Michael Kölling (citado en Utting *et al.*, 2010), no se presenta un rango especial de edades para la enseñanza de la programación en general, incluso asegura que entre más temprano se enseñe, mucho mejor. Sin embargo, afirmó también que se generan "fronteras" con respecto a sus desarrollos cognitivos para poder comprender conceptos y tecnologías específicas, dado que se reflejan problemas en la comprensión de sintaxis complejas por parte de niños pequeños (10 años), y una vez que desorganizan el código no son capaces de repararlo fácilmente. A pesar de que los estudiantes de estas edades pudieron entender los conceptos de programación, no lograron lidiar con sintaxis complejas. Sipitakiat y Nusen (2012) intentaron en su experimento explicar las operaciones básicas de un nuevo entorno llamado Robo-Blocks a niños de 5 a 12 años, pero se dieron cuenta de que los menores de 8 años a menudo tenían dificultad para entender estas operaciones, en particular por los comandos de girar o desplazarse, pues las órdenes que se usaron eran confusas. Estas razones permiten formalizar el rango ideal de edades en los estudiantes para poder iniciar el modelo que nos proponemos, el cual se establece en 10 años, que es la edad promedio de los estudiantes de quinto grado de educación básica primaria.

Realizando una revisión en Internet sobre la experiencia desarrollada por la Fundación Gabriel Piedrahita Uribe, se encontró el contenido temático que se aplicó para el quinto grado de primaria en el Instituto de Nuestra Señora de la Asunción (INSA, 2012). Teniendo en cuenta que esta experiencia se fundamenta particularmente en el uso del entorno Scratch, se indicaron únicamente los temas encontrados en las investigaciones previas:

- Utilización de operaciones matemáticas y booleanas
- Creación y utilización de variables

- Concepto de algoritmo
- Determinación de pasos para resolver problemas
- Uso de identificadores
- Asignación
- Pseudocódigo
- Diagramas de flujo y reglas de construcción
- Operadores y expresiones
- Contadores y acumuladores
- Estructuras condicionales
- Ciclos

De acuerdo con el currículo revisado, estos conceptos se presentaron en este orden, encajando en los temas recopilados en este informe. Dado el corto tiempo con el que se podría contar para la enseñanza de la programación para niños, no sería posible cubrir todos estos temas en la metodología que se pretende plantear. Además, sería necesario revisarlos dependiendo de la herramienta que se seleccione y que permita presentar los temas requeridos.

Por otra parte, se debe determinar un contexto apropiado para aplicar estos nuevos conceptos, enfocado principalmente a la resolución de problemas. Grover (2009) afirmó que ésta, junto con el pensamiento crítico y la gestión de la información son reforzados a través de la profundización de estos temas que integran las ciencias de la computación, y que lo interesante de ellos es que están soportados sobre bases matemáticas, particularmente de niveles que se ven en los colegios. Con base en la revisión del estado del arte, uno de los contextos encontrados con más frecuencia fue la aplicación de la programación en matemáticas. De acuerdo con los estándares de competencias para esta área, establecidos por el Ministerio de Educación Nacional (2006) para el quinto grado, los estudiantes de este nivel pueden construir diferentes tipos de modelos matemáticos y generar diversos métodos para la resolución de problemas sobre los siguientes temas:

- Propiedades de objetos (atributos)
- Estimaciones de medidas
- Operaciones con magnitudes
- Datos para estadística
- Probabilidades de ocurrencia
- Patrones matemáticos y secuencias

- Expresiones numéricas
- Propiedades de figuras geométricas
- Fracciones y notación decimal
- Aplicación de propiedades de números naturales
- Proporcionalidades directas e inversas
- Estrategias de cálculo para resolución de problemas
- Potenciación y radicación
- Objetos bidimensionales y tridimensionales
- Ángulos en diversos contextos
- Sistemas de coordenadas y localización espacial
- Congruencia y semejanza
- Descomposición de sólidos

Precisamente el objetivo fundamental del experimento a plantear es la generación del pensamiento algorítmico para la resolución de problemas matemáticos que involucren como primera medida estos temas. Según Grover (2009), dicho razonamiento ayuda a los estudiantes a establecer una serie de pasos de un problema y plasmarlos en un plan (que puede representarse en un programa, permitiendo que el alumno adquiera habilidades adicionales para definir y establecer claramente el problema, romperlo en subproblemas más pequeños y manejables, y describir la solución en un conjunto bien definido de pasos).

Herramientas para la enseñanza de la programación en niños

En la revisión del estado del arte se encontró un sinnúmero de herramientas para la enseñanza de la programación para niños, cada una con características muy diversas. Tarkan *et al.* (2010), por ejemplo, señalaron una clasificación inicial de herramientas de programación para la enseñanza, dividiéndolas en dos grandes categorías. Una sobre lenguajes de programación visuales, es decir, ambientes gráficos que se pueden usar para crear historias interactivas y juegos entre muchos otros; dentro de éstas se encuentran los lenguajes escritos mediante bloques de construcción (por ejemplo, Alice o Scratch) o través de una sintaxis sencilla basada en reglas (como ToonTalk, Hands o Kodu). En la otra categoría se encontraron los sistemas de programación tangibles, como bloques (Zuckerman, Arida y Resnick, 2005; Sipitakiat y Nusen, 2012) o cartas (Valente, 2004).

Kelleher y Pausch (2005) propusieron una especie de clasificación taxonómica de estas herramientas. Lo interesante de este estudio ha sido la categorización respecto de diversos atributos con los que cuenta cada herramienta según los criterios de enseñanza que se deben manejar, éstos fueron los siguientes:

- Estilo de programación (procedimental –estructural–, funcional, orientado a objetos, basado en objetos, orientado a eventos, basado tanto en eventos como en estado de máquina).
- Constructos de programación disponibles (dependiendo de la herramienta se brinda la posibilidad de trabajar con diferentes comandos particulares de programación, como creación de variables, generación de ciclos, control de flujo mediante condicionales, entre otros).
- Forma de presentación del código en el entorno de programación (mediante texto, imágenes, diagramas de flujo, animaciones, formularios, máquinas de estado u objetos físicos).
- Forma en la que el usuario debe construir el código (tecleando órdenes, ensamblando gráficos, demostrando acciones mediante una interfaz, seleccionando opciones válidas, llenando datos en formularios, ensamblando objetos físicos).
- Soporte adicional para comprender el comportamiento de sus códigos (mediante una historia de fondo para explicar el funcionamiento, mediante soporte de depuración, escogiendo comandos físicos para actuar en el programa, permitiendo hacer cambios en tiempo de ejecución, generando programas de ejemplo).
- Prevención de errores en la interfaz de programación (mediante la forma de los objetos para conectarlos correctamente, seleccionando un conjunto de opciones válidas basado en su posición actual, usando edición dirigida de sintaxis o suministrando mensajes de error para que el usuario los identifique más rápidamente).
- Intento explícito de hacer el lenguaje más fácil de entender (limitación de comandos, selección por palabras clave, remover puntuaciones innecesarias, usar frases para hacerlo lo más natural posible así como redundancias en el lenguaje).
- Soporte colaborativo (soporte en parejas, soporte multiusuario, compartir resultados).

- Consideración de uso adicional (por diversión y motivación, utilidad del sistema para resolver un problema particular, como sistema educativo para ayudar a la enseñanza).

Lameras *et al*. (2010) generaron otra propuesta de caracterización de herramientas de software para la enseñanza de la programación mediante el uso de lenguajes de programación educativos. Estas características son las siguientes:

- Soporte de orientación a objetos (facilidad de transición a lenguajes actuales y procesos adicionales de abstracción –características y funciones).
- Disponibilidad en forma de un entorno de desarrollo integrado (IDE, por sus siglas en inglés) (presentar un entorno de programación similar a los entornos de programas de alto nivel).
- Soporte de programación visual (contribución de trabajar con imágenes para disminuir o eliminar errores de sintaxis).
- Gratuito y de código abierto (con el fin de disminuir costos educativos y para adaptar las herramientas a diferentes contextos o idiomas).
- Desarrollos en comunidades activas.
- Existencia de versiones estables.
- Rangos de edades indicados.
- Idiomas que soporta.
- Soporte de localización.
- Sistema operativo.
- Documentación disponible.

En varias investigaciones se aborda la necesidad de trabajar con lenguajes de programación especialmente diseñados para la enseñanza de la programación. Por ejemplo, Schwartz, Stagner y Morrison (2006) propusieron un lenguaje para niños (*Kid's Programming Language,* KPL) y criterios especiales llamados puntos de diseño, mismos que tuvieron en cuenta para crearlo. Entre ellos se encuentran la diversión que genera, la accesibilidad y usabilidad, la atracción, la simplicidad, la recompensa al presentar resultados inmediatos, su funcionalidad máxima usando mínima codificación, su capacidad de progresividad para la enseñanza, la adecuada preparación que genera para permitir conocer otro lenguaje de programación, la con-

sistencia con diseños modernos de software, la capacidad de ser publicable, la soportabilidad y la posibilidad de internacionalización.

Para esta investigación se recomendaría, con el fin de no desviar el objetivo fundamental de generar una habilidad de resolución de problemas, tener en cuenta un aspecto muy importante como la sencillez de la sintaxis, por encima de un ambiente gráfico. Muchos críticos aseguran que no se presenta un desarrollo claro de las habilidades de los niños para resolver problemas luego de ser sometidos a un curso de programación, puesto que no entienden cómo solucionarlos usando estructuras de programas dado que se estancan en errores sintácticos y no se ven claramente los beneficios educativos (Sipitakiat y Nusen, 2012). Según afirman Fidge y Teague (2009), las personas que incursionan en la programación tienen dificultades en desarrollar un problema algorítmico tratando de dar soluciones a errores y restricciones sintácticas usando los lenguajes formales de programación, pudiendo llegar a causar, además de una pérdida de su objetivo inicial, una sensación de frustración que perjudica más que ayudar. De acuerdo con Stephen Cooper –citado en Utting *et al*., (2010)– gran parte de las primeras frustraciones asociadas con la depuración (solución de errores) se minimizan utilizando un entorno fácil de trabajar, pues así los estudiantes son capaces de avanzar por su cuenta, en su intento de resolver problemas específicos. Fidge y Teague (2009) abordan la programación "libre de sintaxis" con el fin de separar la noción de codificación de la habilidad para resolver problemas mediante programación.

Esta revisión de investigaciones ha arrojado como resultado varias características importantes que debería tener la o las herramientas a ser utilizadas en el curso propuesto:

* Debe soportar las primeras estructuras de programación para su enseñanza (creación de variables, generación de ciclos, control de flujo mediante condicionales, entre otros).
* Debe presentar el código en una sintaxis muy fácil de aprender, y de ser posible que presente un método gráfico para armar o representar el comportamiento del programa por medio de imágenes.
* No necesariamente debe prevenir errores durante su programación, sino más bien debe poder permitir depurar código de una forma sencilla –ya sea mostrando mensajes de error explicando sobre el error y la posible forma de solucionarlo.

* Debe presentar un buen soporte a usuario, con el fin que docentes y estudiantes puedan obtener documentación adicional que le sirva de aporte para la creación de proyectos más complejos.

Dada la restricción de edades y la competencia actual de los niños en cuanto a su experiencia con otros idiomas, se recomendaría que la interfaz (y si es posible, la sintaxis de programación) sea en español. Por lo que se pudo encontrar en esta revisión, no es necesario sujetarse a uno de los lenguajes tradicionales de programación usados para desarrollar aplicaciones software, sino que se podría trabajar con herramientas que permitan usar un pseudolenguaje para que ejecute órdenes y que admitan las primeras estructuras de programación para lograr una iniciación de los niños en la enseñanza de la programación. Tal es el caso de herramientas como Blockly (Google Project Hosting, 2011), Scratch (MIT Media Lab, 2013), Light-Bot (Yaroslavski, 2013), PiktoMir (Rogozhkina y Kushnirenko, 2011), Gidget (Lee y Ko, 2011), RoboMind (Research Kitchen-Universiteit Van Amsterdam, 2014), Squeak Etoys (Viewpoints Research Institute, 2014), entre muchos otros.

Técnicas de aplicación de la enseñanza de la programación en el aula

El desarrollo del aprendizaje se entiende mejor a través de la teoría cognitiva planteada por Piaget (Kliegman *et al.*, 2011). Él describe cómo los niños construyen activamente el conocimiento por sí mismos a través de procesos vinculados de asimilación (tomando nuevas experiencias de acuerdo a esquemas existentes) y acomodación (creando nuevos patrones de entendimiento para adaptarse a la nueva información), y de esta forma, los niños están reorganizando los procesos cognitivos en forma continua y activa. Los conceptos básicos de la teoría cognitiva de Piaget se han mantenido y se han utilizado como base para la creación de otras de aprendizaje, como el construccionismo, propuesto por Seymour Papert. De acuerdo con Papert (1980), el aprendizaje de un nuevo concepto se vuelve fácil si se logra asimilar la nueva idea aprendida comparándola y acoplándola a la colección de los modelos existentes en su conocimiento previo; si no se logra, el aprendizaje se torna difícil. En otras palabras, lo que un individuo puede aprender y cómo lo aprende depende, principalmente, de los modelos que tiene disponibles en su mente. Papert ha trabajado con este

modelo pedagógico en la enseñanza usando un pseudolenguaje de programación llamado LOGO y ha sido un gran promotor de llevarlo las aulas escolares como base para enseñar matemáticas a los niños. Su trabajo le ha permitido demostrar que no solo es esencial la enseñanza de conceptos bajo un contexto sino que también es muy importante el uso de un soporte externo adecuado para ampliar las potencialidades de la mente humana en cualquier nivel de su desarrollo (Ackermann, 2001).

Como se pudo observar en la revisión de literatura, el aprendizaje por construcción ha sido aplicado exitosamente en muchos casos, pudiendo perfectamente ser utilizado para la enseñanza de los nuevos conceptos que involucra la generación del pensamiento algorítmico. La edad apropiada para aprenderlos y que los niños puedan no solo generar sino también mejorar sus habilidades se encuentra dentro de los 8 a los 11 años de edad. Según Warren Buckleitner (citado en Walton-Hadlock, 2008), las tecnologías apropiadas que sean utilizadas en los estudiantes de educación básica deberían estimularlos a generar socialización, a expandir sus habilidades y conocimientos y suministrándoles múltiples niveles de reto, que sean sencillos de utilizar y que puedan fácilmente ser enlazados con otros tipos de medios, como libros o tutoriales.

El modelo que se pretende generar tomando como base esta revisión de literatura debe estar basado en el desarrollo del pensamiento algorítmico, con el fin de que los niños puedan generar nuevas estructuras de aprendizaje aplicables en el área de matemáticas para la resolución de problemas. Mientras que Grover (2009) lo llamó pensamiento algorítmico, en la National Academy of Sciences (2010) se le denominó pensamiento computacional, el cual "debería ser concebido incluso como una habilidad intelectual fundamental comparable a la lectura, la escritura, la narración y la aritmética", es decir, debe ser una habilidad cognitiva que una persona debería tener en esta época actual.

Para hacer que los niños desarrollen estas habilidades en forma eficaz, sería necesario generar un curso de programación bien estructurado y diseñado, enfocado particularmente a desarrollar las habilidades de los estudiantes para aplicar su conocimiento a la solución de problemas reales y, al mismo tiempo, generar programas coherentemente desarrollados (Lameras *et al.*, 2010). Resnick *et al.* (2009) señalan tres habilidades que se adquieren en la enseñanza de la programación: para pensar creativamente, para razonar sistemáticamente y para trabajar colaborativamente;

no se busca preparar a las personas para que se conviertan en profesionales programadores de software sino para nutrir a una nueva generación de pensadores sistemáticos y creativos mediante el uso de la programación para expresar sus ideas. Uno de los retos que se presentan en la enseñanza de la programación es justamente generar el interés por este aprendizaje, balanceando los aspectos educativos y motivacionales mediante la alineación de materiales interesantes de trabajo con marcos de trabajo educativos bien fundamentados (Repenning y Ioannidou, 2008). Una de las formas más usadas para generar este interés por el aprendizaje está en el uso de juegos para la enseñanza. De acuerdo con Kirriemuir y McFarlane (2004), se puede generar un gran interés en el que incluso los estudiantes se vuelvan ajenos a las distracciones, y esto se logra durante actividades que sean muy divertidas, como en el uso de los videojuegos ya que "al combinarlo además con la curiosidad y la fantasía, adquieren un nivel de compromiso tal que desaparecen las distracciones".

Se encontró una amplia documentación sobe el beneficio del uso de los videojuegos en la enseñanza, al igual que una extensa información respecto de sus inconvenientes. Mitchell y Savill-Smith (2004), en su estado del arte, encontraron que el uso de los videojuegos puede estimular la motivación y el compromiso de los usuarios por la consecución de objetivos mediante el desarrollo y el aprendizaje de ciertas habilidades sociales y cognitivas. Sin embargo, hallaron también que su uso frecuente, especialmente los juegos que contienen algún contenido violento, pueden generar sentimientos de ansiedad y tendencias psicosociales negativas como violencia o aislamiento así como problemas de salud si su uso llegase a ser catalogado como adictivo. Una forma para poder utilizarlos es su adecuación al contexto actual de trabajo en el grupo. Malone (citado en Kirriemuir y McFarlane, 2004:4) indica que para lograr el uso efectivo de un juego contextualizado en una experiencia de aprendizaje, las actividades propuestas a los estudiantes deberían ser estructuradas de tal forma que puedan incrementar o disminuir el nivel de dificultad del videojuego utilizado, con el fin de trabajar específicamente en el nivel de habilidad que posee, y de esta manera evitar sentimientos de ansiedad, frustración o aburrimiento.

Otra forma utilizada para generar interés en el aprendizaje fue el presentado por Hug, Guenther y Wenk (2013), cuyo proyecto logró el interés en los estudiantes presentándoles una oportunidad de construir proyectos de la vida real para solucionar problemas específicos, pero sin salirse del diseño

curricular que se les ofrecía. Proyectos de este tipo deberían ser diseñados con sumo cuidado y sin improvisaciones, con el fin de poder orientar de una manera adecuada al estudiante sin que éste caiga en sentimientos de frustración al no lograr los objetivos, o incluso que deba recurrir a técnicas como el ensayo y error para el desarrollo de su proyecto. Según Stephen Cooper (citado en Utting *et al*., 2010), el tipo de estrategia basado en ensayo y error no es conveniente trabajarlo frecuentemente puesto que no genera una construcción sólida de conocimiento y es mejor que los estudiantes desarrollen otras estrategias exitosas hacia la resolución de problemas. Para el diseño curricular del curso, se podría plantear, entonces, el uso de la matemática para solucionar problemas reales en diferentes contextos, preparados con cuidado de acuerdo con la temática a trabajar en cada sesión y pudiendo detectar algunas de las múltiples posibilidades de resolución que se generarán en el desarrollo del proyecto, durante la enseñanza de los nuevos modelos algorítmicos que se presenten.

Igualmente, existen otras formas que han sido exitosas en la enseñanza por medio de tecnologías de la información, no solo aplicadas en la programación, sino también en otros contextos. Por ejemplo, Wen-Yu Lee y Tsai (2013), en su revisión de literatura, encontraron varios tipos de técnicas de enseñanza que implicaban el uso de tecnologías de la computación, especialmente para fomentar habilidades de orden superior como las de resolución de problemas o incluso el pensamiento crítico, tales como el uso de simulaciones, materiales multimedia, sistemas integrados, cursos mixtos, tutoriales y evaluaciones asistidas por computador, entre otras. Sin embargo, estas técnicas de enseñanza requerirían mayores esfuerzos interdisciplinarios y tiempos de preparación curricular. Uno de los aspectos a tener en cuenta para la aplicación de este tipo de proyectos, que integren la enseñanza de la programación en un entorno académico, son las restricciones que se pudieran presentar, ya que se debe buscar no solo que existan facilidades en su desarrollo sino también que tanto el docente como los estudiantes no pierdan su motivación en explorar esta nueva temática con la aplicación de una tecnología que sería novedosa para ambos actores. En un informe presentado por Osborne y Hennessy (2003), se encontró que dentro de las restricciones más importantes se encuentran la falta de tiempo –tanto por parte del docente como del estudiante– para adquirir confianza y experiencia con el uso de la tecnología, el acceso limitado a recursos, un currículo adicional sobrecargado con contenido que no esté enfocado

correctamente al uso de la tecnología, y la falta de una guía específica para utilizar la tecnología para soportar adecuadamente el aprendizaje.

Teniendo en cuenta estos conceptos, se realizó una clasificación de las 18 experiencias encontradas en la enseñanza de la programación en niños y jóvenes, sintetizada en el cuadro 2 y que posteriormente describimos.

CUADRO 2
Clasificación de experiencias encontradas en la enseñanza de programación en niños y jóvenes

Clasificación	**Experiencias en que se aplicó**
Uso de videojuegos para la enseñanza	Rogozhkina y Kushnirenko, 2011; Lee y Ko, 2011
Creación de proyectos con temática fija	Liu, Cheng y Huang, 2011; Valente, 2004; Sipitakiat y Nusen, 2012; Zuckerman, Arida y Resnick, 2005; Tarkan *et al.*, 2010
Creación de entornos para generación de situaciones	Felleisen, *et al.*, 2009; Wolz *et al.*, 2009; Burke y Kafai, 2010; Kelleher, Pausch y Kiesler, 2007; Meyers *et al.*, 2009; Rizvi *et al.*., 2011
Creación de videojuegos	Doerschuk, Liu y Mann, 2012; Javidi y Sheybani, 2009; Rodger *et al.*, 2010
Herramientas computacionales como apoyo a la resolución de problemas	Abramovich, 2013
Generación de torneos o competencias	Tomcsányiová y Tomcsányi, 2011

Elaboración propia.

Uso de videojuegos para la enseñanza

La primera categoría se encontró en experiencias que generan interés por el aprendizaje de la programación mediante el uso de videojuegos. En esta técnica se clasificó el uso de elementos computacionales tipo juego que presentaran un objetivo particular, una historia de fondo y un conjunto de reglas y de niveles a cumplir. Dentro de estas experiencias se ubican la de Rogozhkina y Kushnirenko (2011) –que usa un entorno de juego llamado PiktoMir– y la de Lee y Ko (2011) con un juego llamado Gidget.

El objetivo del experimento de Rogozhkina y Kushnirenko (2011) fue determinar la factibilidad de la herramienta para enseñar los primeros elementos de programación a niños pequeños, dado que para ello se requiere tradicionalmente de ambientes de lecto-escritura, estableciendo un rango de edad limitado para que los infantes estén en posibilidad de aprender. Para probar su hipótesis, los autores trabajaron con 42 niños de entre 5 y 7 años de una escuela en Moscú, con poca habilidad para la lectura y la escritura; utilizaron el programa PiktoMir, un juego que presenta un ambiente sin texto y que brinda soporte a la enseñanza de la programación; se trata de un pequeño robot, llamado Fidget, que debe arreglar recubrimientos dañados en el piso, y al cual se le deben asignar las órdenes dado que no puede tomar decisiones por sí mismo. El experimento se realizó durante ocho semanas, con dos jornadas cada una y una duración de 20 minutos. En la primera sesión se enseñaba la teoría del juego sin el uso del computador y en la segunda se realizaban ejercicios de programación. Durante la ejecución del experimento se tomaba un registro del material de enseñanza, procedimientos de clase, retroalimentaciones y observaciones generales. Al completar el curso, los niños fueron sometidos a un examen final, en el que se presentaban tres bloques de tareas, de 6 actividades cada uno; el estudiante debía dibujar el desplazamiento del robot después de presentar un conjunto de instrucciones o encontrar el error de una serie de instrucciones mostrando el movimiento del robot. Al final, casi todos los niños (41 de 42) habían comprendido cómo escribir instrucciones secuenciales usando elementos gráficos y 75% de ellos aprendieron los conceptos de ciclos y subrutinas para aplicarlos en sus programas.

La experiencia de Lee y Ko (2011) tuvo como objetivo mostrar una mejora motivacional en la programación para un aprendizaje exitoso, utilizaron una herramienta que brindara una retroalimentación más adecuada de los posibles errores sin el uso de mensajes de fallo en un programa. Para el proyecto se reclutaron inicialmente 250 personas de todo el mundo a través de un sistema llamado Amazon Mechanical Turk. Se presentaron sujetos de 18 a 59 años para jugar una aplicación llamada *Gidget*, desarrollada en HTML5 y JavaScript mediante jQuery, la cual se guiaba a través de niveles que enseñaban el diseño y el análisis de los algoritmos usados por un robot; éste debía ayudar a limpiar un derrame de químicos en un terreno con el fin de proteger la vida silvestre, desplazándose para encontrar estos elementos, revisando sus características y moviéndolos a otro

lugar. Para el experimento se requirió que los participantes completaran un conjunto de 15 niveles en el juego, y se compararon los resultados de quienes recibieron una retroalimentación más adecuada de los errores con los de aquellos que no recibieron retroalimentación suficiente. Teniendo en cuenta que 116 de los 250 eran completamente novatos en temas de programación, se observó que se obtuvieron los mismos resultados que los que contaban con conceptos previos al respecto.

Creación de proyectos con temática fija

Dentro de las experiencias revisadas en el estado del arte, se encontró una técnica interesante que consiste en crear diversos tipos de proyectos usando simuladores o aplicaciones que admitan manipular el entorno de trabajo para generar diversas situaciones, permitiendo plantear problemas particulares que puedan resolverse a través del razonamiento lógico aplicando procedimientos algorítmicos. Estas técnicas se clasificaron dependiendo del sistema utilizado (ya sea hardware especializado, software o una combinación de ambos). Además, se tuvo en cuenta que los experimentos presentados mostraban una limitante de trabajo hacia una temática particular. Dentro de estas investigaciones se encontraron el uso de cartas computacionales de Valente (2004), los sistemas de bloques de Sipitakiat y Nusen (2012) y de Zuckerman, Arida y Resnick (2005); el trabajo de Liu, Cheng y Huang (2011) usando el simulador Train B&P y la experiencia de creación de recetas de (Tarkan *et al*. 2010) con el programa Toque.

El experimento de las cartas computacionales de Valente (2004) se basó en una propuesta de un curso exploratorio para niños con el fin de enseñar programación de una manera teórica, dado que los algoritmos y la programación han sido considerados a menudo como campos muy formales o complejos y se ha buscado desde hace tiempo representarlos de una forma más sencilla, a través de objetos físicos e interacciones con otros. El objetivo de ese artículo fue presentar el curso propuesto para la enseñanza de la programación. Cada carta se comportaba como un elemento sobre el cual se transportaba un elemento computacional definido, construido y transportado dependiendo de la colocación de las cartas y de un conjunto de reglas definidas. Cada carta se conformaba de una pequeña pieza cuadrada de papel con un dibujo en su parte superior y con indicadores de entrada y salida llamados puertos. Luego de armar un circuito con las

piezas cuadradas, se procedía a utilizar un indicador que debía desplazarse sobre estas cartas, respetando las reglas.

Los trabajos de Sipitakiat y Nusen (2012) y de Zuckerman, Arida y Resnick (2005) presentaron sistemas de programación tangibles –que se han convertido en una nueva tendencia en la enseñanza de esta actividad–, permitiendo que los estudiantes crearan un programa conectando bloques físicos con componentes electrónicos, que generaban un conjunto de instrucciones, y que debían combinarse para controlar los movimientos de un robot. En el experimento de Sipitakiat y Nusen (2012) se trabajó con el sistema RoboBlocks, participaron 52 niños entre 8 y 9 años cuyas actividades implicaban mover el robot entre un laberinto o realizar un gráfico en el suelo. Cuando no se obtenía el resultado esperado se recurrió a un proceso de tres formas de depuración con los estudiantes: la revisión paso a paso, la inserción de banderas en el lugar del problema y el transportador (medidor de giro) para el robot. Un hallazgo de este experimento fue que aun cuando al principio fue bastante motivador para los niños, el interés se perdía al no presentar actividades que brindaran una solución a problemas diferentes a los movimientos restringidos de los robots.

El experimento de Zuckerman, Arida y Resnick (2005) utilizó el ensamble de bloques electrónicos para encender luces o emitir sonidos. En este caso, se les catalogó como manipulativos tipo Montessori porque estos bloques permitían generar un mejor modelamiento de conceptos más abstractos. Participaron 25 niños de entre 4 y 11 años que desarrollaron un conjunto de proyectos que permitían establecer conceptos más complejos como tasas, acumulación, retroalimentación y probabilidad. Sin embargo, se logró apreciar que aunque se intentó enseñar temas con un alto nivel de abstracción, solo estudiantes de 10 años lograron establecer analogías interesantes dado que hasta esa edad desarrollan la capacidad cognitiva de abstracción.

La experiencia de Liu, Cheng y Huang (2011) se realizó para investigar el efecto de usar simuladores en la resolución de problemas computacionales. Participaron 117 estudiantes de bachillerato que ingresaron a una universidad al noreste de Taiwán, inscritos a la asignatura Introducción a las ciencias de la computación, y que no tenían ninguna experiencia en la programación. La creación y el control de transporte de un sistema de ferrocarril fue el contexto de trabajo para el experimento, dado que podía contar con un

micromundo que permitía a los aprendices ser "arquitectos", generando y probando ideas de construcción, utilizando lo que se conoce como aprendizaje por construcción, el modelo pedagógico presentado por Papert (1980). El programa Train B&P es una simulación de construcción de ferrocarriles, en el que los estudiantes pueden diseñar sistemas de carriles y establecer, mediante programación, los comportamientos del desplazamiento de los trenes en los rieles. Previo a la aplicación del experimento con la herramienta, se les presentó un contenido basado en lecciones teóricas sobre programación básica (algoritmos), y se les hizo una prueba para medir su nivel de "flujo". Luego, los participantes trabajaron durante dos semanas en el uso de esta herramienta, tiempo en el cual se tomaron indicadores adicionales de actividad. Durante este tiempo, a los estudiantes se les enseñaban los conceptos de orientación a objetos, condicionales, iteraciones y comunicaciones entre objetos. Al finalizar el proyecto, se realizó una prueba de experiencia de aprendizaje en la que pudieron obtener impresiones sobre lo que aprendieron con el juego, esperando que tuvieran una visión de cómo esta simulación había generado habilidades de resolución de problemas computacionales obtenidos de los registros de actividad y de las pruebas.

El trabajo de Tarkan *et al*. (2010) se fundamentó en mostrar la importancia del uso de la narración y la generación de argumentos para la enseñanza de la programación con el fin de superar limitaciones impuestas por la aplicación. Para ello los investigadores, junto con un grupo de niños, desarrollaron un lenguaje llamado Toque, implementándolo en un escenario de cocina para crear programas tipo "recetas" con animaciones controladas con un dispositivo de detección de movimiento. Participaron nueve niños de 7 a 11 años y el programa se desarrolló durante dos años, los pequeños, además, jugaban con su propia creación para determinar aspectos por mejorar y trabajar proyectos al respecto.

Creación de entornos para generación de situaciones

Se encontró una técnica en la que se utiliza una herramienta de desarrollo para crear un entorno particular desde cero y por medio de procesamientos secuenciales lógicos y/o matemáticos se establecen situaciones interesantes las cuales los niños pueden explorar y manipular mediante la programación. Entre las experiencias que utilizaron esta técnica se encontraron los trabajos de: Felleisen *et al.*, (2009), mediante el uso del entorno de programación Dr. Scheme; de Wolz *et al*. (2009), Burke y Kafai (2010) y

Rizvi *et al*., (2011), que usaron el entorno de programación Scratch; de Kelleher, Pausch y Kiesler (2007) con el entorno Alice; y el de Meyers *et al*., (2009) con el entorno de programación Processing.

Felleisen *et al*. (2009) crearon en su propuesta entornos que requirieron funciones matemáticas para determinar y controlar el comportamiento de cohetes usando Dr. Scheme, un entorno que utiliza el paradigma de programación funcional. En dicho experimento se buscó proveer un currículo que presentara una forma de enseñar conceptos pre-algebraicos, trabajando en pruebas de teorías matemáticas y diseño orientado a objetos. En este proyecto participaron estudiantes de educación media (de 10 a 14 años), y han trabajado con estos grupos desde 2006 creando un conjunto de proyectos tipo juego e implementando un curso especial de nueve semanas de duración con sesiones de dos horas semanales. Los autores ya habían trabajado con este entorno de programación desde 2003 con estudiantes universitarios, realizando otro tipo de proyectos.

Wolz *et al*., (2009) utilizaron el entorno Scratch para crear todo tipo de entornos 2D que involucraran procesos asincrónicos y concurrentes, como videojuegos, historietas, tarjetas animadas, parodias de televisión, tutoriales educativos, simulaciones científicas, entre otros; la investigación se realizó con estudiantes que iniciaban el curso de Ciencias de la computación en la universidad. Los autores presentaron una justificación de integración del trabajo con Scratch en el currículo de Ciencias de la computación para los primeros grados de universidad, basados en la capacidad de generar una transición más amigable en la enseñanza de la programación para lenguajes más robustos como Java o C.

Rizvi *et al*., (2011) utilizaron el entorno Scratch para crear animaciones, proyectos multimedia y juegos con el fin de aplicarlo en el curso inicial de Ciencias de la computación y determinar su eficacia en la mejora de la retención, el rendimiento y la actitud en cuanto a habilidades de programación. En esta experiencia participaron alumnos que recién se matricularon a la universidad y tomaron el curso referido. En particular, centraron su atención en estudiantes "en riesgo", es decir, con una preparación muy débil en matemáticas. Con ellos se buscó trabajar en la elaboración de varios proyectos multimedia, los cuales involucraron el uso de una alta cantidad de gráficos, elementos animados y sonidos.

Burke y Kafai (2010) también usaron Scratch, en este caso con jóvenes de último grado de bachillerato (K-12), con el fin de crear historias y

determinar si la creación de programas de computador podría ayudarles a desarrollar su capacidad de narrar cuentos y mejorar sus habilidades de escritura. Aunque se trabajó en la enseñanza de los conceptos básicos de la programación, se observó que durante seis semanas el experimento se enfocó más en los conceptos usados en el desarrollo de las historias que en los componentes técnicos de la generación de los programas.

Kelleher, Pausch y Kiesler (2007) trabajaron también en la creación de historias usando la herramienta Alice. En este experimento participaron 88 niñas de tropas Scout de un promedio de 12.5 años, con el fin de demostrar que se puede desarrollar un sistema de enseñanza de la programación para niños, en particular para niñas, presentándoles una experiencia positiva con la programación de computadores, inspirando a más estudiantes a continuar con este tipo de asignaturas de Ciencias de la computación e incluso mejorar la representación de las mujeres en esta área. Los autores habían trabajado con la herramienta Alice desde 2005.

Meyers *et al.*, (2009) usaron la herramienta Processing para enseñar Ciencias de la computación mediante la creación de música computacional. En esta experiencia se trabajó con niños de los grados 6° a 12° (entre 12 y 17 años) con cierta experiencia musical y habilidad matemática. Se realizó un curso intensivo de 16 días, con sesiones entre semana de dos horas y media y los sábados durante una jornada de siete horas.

Creación de videojuegos

Otra técnica muy interesante es el uso de herramientas para desarrollar videojuegos. Se asemeja mucho a la técnica anterior en cuanto a que se debe crear un entorno desde cero además de generar procesamientos secuenciales lógicos y/o matemáticos para establecer situaciones de juego. Éstas se obtienen mediante una explicación adicional sobre las bases para la creación de juegos. Entre las exeriencias que utilizaron esta técnica se encontraron las presentadas por Doerschuk, Liu y Mann (2012), mediante el entorno de programación Greenfoot; el experimento de Javidi y Sheybani (2009), que trabajaron con dos entornos llamados Kahootz y Squeak, y la experiencia de Rodger *et al.* (2010), quienes enseñaron a desarrollar videojuegos utilizando Alice.

Doerschuk, Liu y Mann (2012) presentaron en su experiencia una forma para incrementar la participación estudiantil en los programas de Ciencias de la computación en la Universidad Lamar en Texas, EUA. Para

ello desarrollaron un plan estratégico llamado INSPIRED, que abordó la enseñanza de la creación de videojuegos a estudiantes de primer semestre. En conjunto con un grupo de jóvenes de último semestre crearon los materiales instruccionales, enseñaron y ayudaron a aplicar este plan. Para ello utilizaron la herramienta Greenfoot, un entorno gratuito para desarrollar videojuegos basado en Java, y prepararon una serie de recursos y ambientes para preestablecer los escenarios, los objetivos y las reglas. A su vez, prepararon el material educativo y los códigos fuente de varios de los juegos que los estudiantes debían modificar, significando que en varios proyectos no partían desde cero sino que aprendían conceptos a partir de recursos y proyectos ya existentes.

Javidi y Sheybani (2009) desarrollaron un plan similar para estudiantes de bachillerato en Virginia, EUA, llamado DIGINSPIRED, cuyo objetivo no solo fue enseñar sobre programación de computadores sino también sobre la creación de gráficos computacionales y animación así como las tecnologías requeridas para desarrollar este tipo de videojuegos. El trabajo consistió en crear juegos cuya temática abordara una de cuatro áreas de ciencias: reciclaje, nutrición, ejercicio físico o consumo de drogas. El proyecto se llevó a cabo con 89 alumnos de tres colegios de bachillerato durante el año, asistiendo a 15 sesiones los sábados y un curso adicional de verano. Se observó que en el desarrollo de videojuego, varios estudiantes demoraron dos semanas en implementarlo.

Rodger *et al.* (2010) integraron el desarrollo de videojuegos en varias áreas en los niveles de educación básica, media y universitaria. En esta experiencia se identificó que con esta técnica los estudiantes de tercero a quinto grados deberían se capaces de crear y seguir un algoritmo para codificar y probar un sencillo programa secuencial, mientras que los de sexto a octavo grados deberían tener la capacidad de codificar y probar programas para resolver un problema usando variables, decisiones y ciclos. Los autores han trabajado en este proyecto durante ocho años a nivel de bachillerato y cuatro en el universitario, tiempo en el que han generado una serie de recursos entre tutoriales y archivos para usar en el desarrollo de los juegos.

Uso de herramientas computacionales como apoyo a la resolución de problemas

En este apartado se encontró la investigación de Abramovich (2013). El autor presentó una compilación de diez reportes de experiencias en va-

rios países que usaron herramientas computacionales para la resolución de problemas. Abramovich argumentó que la efectividad en la enseñanza de las teorías matemáticas depende del uso apropiado de tecnologías para apoyarla, demostrando la dualidad de la enseñanza y el uso de los computadores. En las experiencias presentadas por el autor, se mencionan algunas herramientas de software como GeoGebra, Rhinoceros, VisualMath, Calculadora gráfica, Mathematica, Maple, MiniTab, SimCalc, MathWorlds, SketchPad, Pathom, NonEuclid, entre otras. Éstas permiten generar modelos matemáticos a partir del ingreso de ecuaciones, variables y restricciones con el fin de analizar datos y presentar soluciones basadas en números. Abramovich indica que usando GeoGebra en particular se buscó reformular problemas tradicionales algebraicos en entornos más significativos; con Rhinoceros se mostró el valor del modelado tridimensional mediante el uso de la geometría descriptiva y de ecuaciones matemáticas para crear sólidos de revolución; con VisualMath se demostró la posibilidad de crear libros electrónicos, presentando diversos gráficos interactivos que soportaron exploraciones de conceptos matemáticos particulares. Básicamente se mostró cómo usando estas herramientas, manipulando datos de variables y realizando ciertas codificaciones, se podrían presentar casos de estudio de problemas reales y comprender determinadas situaciones o encontrar resultados adecuados.

Generación de torneos o competencias

En este rubro se encontró una técnica que utiliza torneos competitivos como un método para la enseñanza y para probar el conocimiento adquirido por los niños en el área de informática elemental, una asignatura que se brinda en el currículo de educación primaria en Eslovaquia. Dentro de ella se enseñan conceptos de información, comunicación, TIC, resolución de problemas, pensamiento algorítmico y flujos, entre otros, dando a los estudiantes las bases de la programación y la habilidad para resolver problemas mediante tecnologías digitales. Tomcsányiová y Tomcsányi (2011) prepararon y probaron un torneo especial para niños entre 8 y 9 años, con el fin de generar actividades intra y extraclase en los colegios para mejorar sus habilidades para su participación. Se basaron en un torneo existente –llamado Bebras y que se realiza en Lituania con estudiantes de educación secundaria– para desarrollar el propio llamado Pequeño Bebras. Dentro de las actividades realizadas se encontraron la traducción de tareas, la adap-

tación de los datos, la recategorización de problemas, la simplificación de actividades, el rediseño de imágenes y la implementación de software para generar tareas interactivas. Se observaron otros aspectos más pedagógicos para bajar el complejo nivel de las preguntas, reduciendo los textos, mostrando elementos concretos más que intentar presentar ideas abstractas, adaptando las imágenes a colores y tamaños apropiados, y delimitando la cantidad de tareas para no cansar a los niños.

Modelos de evaluación

En los experimentos de evaluación de Agina (2012) se consideró una serie de variables en la realización de sus pruebas, como la cantidad de ejercicios ejecutados, el tiempo que tomó desarrollar cada uno, el número de respuestas correctas e incorrectas y de tareas que no se pudieron completar, entre otras. En ese mismo experimento se consideró una variable adicional para medir el pensamiento creativo en la resolución de problemas, como la necesidad de los estudiantes de solicitar ayuda extra al docente para explicar un problema particular de la prueba. Otra de las variables adicionales utilizadas tuvo que ver con la sensación de satisfacción que causaba la aplicación de las pruebas. Esto se logró mediante una serie de preguntas posteriores, con las cuales se indagaba si éstas eran, para su concepto, entendibles y fáciles de realizar, entre otras. Esta determinación adicional de la sensación que causa esta actividad en los estudiantes se vio también en el experimento de Giannakos y Jaccheri (2013), en el que se formularon preguntas para determinar el grado de satisfacción, utilidad y facilidad de las pruebas. Estas variables ayudaron a identificar si los estudiantes habían adquirido intrínsecamente la habilidad para comprender los problemas que se les presentaron en la prueba. Liu, Cheng y Huang (2011) hablaron sobre esta sensación de satisfacción como un estado mental, conocido como "flujo", en el que puede estar un estudiante con el fin de generar mejores aprendizajes y formas de resolver problemas. Este mismo aspecto lo refirieron Kirriemuir y McFarlane (2004), quienes argumentaron que es en este estado de "flujo", en el que se genera un mayor interés por parte de los estudiantes haciendo que incluso se vuelvan ajenos a las distracciones. Liu, Cheng y Huang (2011) presentaron una forma de medir este estado basado en el modelo de flujo de tres canales (aburrimiento, ansiedad, estado de flujo) al determinar un balance entre niveles de habilidades percibidas y retos percibidos.

Basados en los experimentos de Agina (2012), se podría plantear un diseño de evaluación que cuente con preguntas tanto de tipo matemático para resolver problemas sobre una temática recientemente vista, como adicionales sobre las opiniones que los estudiantes tienen sobre la prueba. Esta evaluación debería estar basada en lo que normalmente ven los alumnos en el área de matemáticas, prácticamente de la misma forma a una prueba tradicional que desarrollaría el docente en su área. Podría tener las siguientes variables:

- Cantidad de ejercicios ejecutados y cantidad de ejercicios que no pudo completar.
- Tiempo que le tomó desarrollar los ejercicios.
- Procedimiento de resolución del ejercicio.
- Número de respuestas correctas e incorrectas.
- Cantidad de solicitudes de ayuda extra por parte del docente para explicar un ejercicio.
- Nivel de satisfacción que causó la aplicación de las pruebas.

Esta última pregunta cubriría una variable adicional, la cual se observó en el experimento de Giannakos y Jaccheri (2013); en él se realizaban preguntas para determinar el grado de satisfacción, utilidad y facilidad de las pruebas. Esto ayudaría en parte a determinar si se ha adquirido intrínsecamente la habilidad para comprender los problemas que se presentaron en la prueba, además serviría de retroalimentación para mejorar las preguntas si se logra encontrar que no generan un nivel de satisfacción en forma general.

Conclusiones

Mediante la revisión del estado del arte en cuanto a experiencias significativas de enseñanza en lo referente a la comprensión y el desarrollo de los primeros algoritmos se lograron determinar importantes aspectos. En cuanto a las temáticas y contextos adecuados de enseñanza de la programación para niños, se establece un conjunto de temas en las áreas de matemáticas, estadística y geometría que podrían ser abordados mediante aplicaciones prácticas algorítmicas, utilizando para ello varias estructuras de programación a nivel básico, como el uso de variables y diversas estructuras de control. Sobre las características de herramientas adecuadas para la programación en niños, aunque es obviamente necesario que éstas pue-

dan cubrir las estructuras que se pretenden enseñar, es también necesario contar con una aplicación cuyo formato de código tenga una sintaxis muy fácil de entender, con una interfaz en el idioma nativo donde se aplique (en este caso español), de ser posible que cuente con un modo gráfico y que permita depurar código de una forma sencilla, mostrando mensajes de error explicando sobre el mismo y la posible forma de generar una solución, con el fin de no desviar la atención en lo realmente importante como es el desarrollo del pensamiento algorítmico. Respecto de las técnicas de aplicación de la enseñanza de la programación en el aula, se observa que la generación de proyectos que planteen situaciones particulares y el uso de los juegos con objetivos, reglas y niveles definidos y enlazados mediante una historia de fondo, entre otras, son adecuadas para brindar una enriquecedora experiencia de enseñanza de la programación en los niños. A su vez, debe pensarse en la generación de un curso apropiado para niños entre 8 y 11 años, edad en la que cuentan ya con la habilidad de comprender procesos abstractos en forma sistemática y creativa.

Finalmente, se logró determinar un conjunto de características para la evaluación de resultados, que permitan medir el mejoramiento de las competencias de resolución de problemas matemáticos y así establecer la efectividad del modelo que se proponga. Como trabajo futuro, se plantea utilizar estas líneas guía para realizar una selección adecuada de las técnicas, herramientas y temáticas con el fin de elaborar un plan curricular adecuado al área de tecnología para el nivel 5° de educación básica primaria, que pueda ser implantado en una institución académica y medir su efectividad en el mejoramiento de las habilidades matemáticas en los niños.

Referencias

Abelson, H.; Sussman, G. J. y Sussman, J. (1996). *Structure and interpretation of computer programs*, Boston: The MIT Press.

Abramovich, S. (2013). "Computers in mathematics education: An introduction", *Computers in the Schools,* vol. 30, núms. 1-2, pp. 4-11 (doi:10.1080/07380569.2013.765305).

Ackermann, E. (2001). "Piaget's Constructivism, Papert's Constructionism: What's the difference?", *Future of Learning Group Publication*, vol. 5, núm. 3, p. 438. Disponible en: http://learning.media.mit.edu/content/publications/EA.Piaget%20_%20Papert.pdf (consultado el 15 de septiembre de 2014).

Agina, A. (2012). "The effect of Nonhuman's external regulation on young children's creative thinking and thinking aloud verbalization during learning mathematical tasks", *Computers in Human Behavior*, vol. 28, núm. 4, pp. 1213-1226 (DOI:10.1016/j.chb.2012.01.022).

Alt, C.; Astrachan, O.; Forbes, J.; Lucic, R. y Rodger, S. (2006). "Social networks generate interest in computer science", en *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, Nueva York: ACM, pp. 438-442 (doi:10.1145/1121341.1121477).

Burke, Q. y Kafai, Y. B. (2010). "Programming & storytelling: Opportunities for learning about coding & composition", en *Proceedings of the 9th International Conference on Interaction Design and Children*, Barcelona: ACM, pp. 348-351 (DOI:10.1145/1810543.1810611).

Code.org. (2013). *Code.org* (página web). Disponible en: http://code.org/

Delgado, J.; Güell, J.; García, J.; Conde, M. y Casado, V. (2013). "Aprendizaje de la programación en el Citilab", *Revista Iberoamericana de Ciencia, Tecnología y Sociedad*, vol. 8, núm. 23, pp. 123-133.

Doerschuk, P.; Liu, J. y Mann, J. (2012). "An INSPIRED game programming academy for high school students", *Proceedings-Frontiers in Education Conference, FIE*, Washington, DC: IEEE Computer Society, pp. 1-6 (doi:10.1109/FIE.2012.6462240).

Felleisen, M.; Findler, R.; Flatt, M. y Krishnamurthi, S. (2009). "A functional I/O system: Or, fun for freshman kids", *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, Nueva York: ACM, pp. 47-58 (doi:10.1145/1631687.1596561)

Fidge, C. y Teague, D. (2009). "Losing their marbles: Syntax-free programming for assessing problem-solving skills", *Proceedings of the Eleventh Australasian Conference on Computing Education, 95*, Wellington: Australian Computer Society, Inc, pp. 75-82.

Giannakos, M. y Jaccheri, L. (2013). "What motivates children to become creators of digital enriched artifacts?", *Proceedings of the 9th ACM Conference on Creativity & Cognition,* Sydney: ACM, pp. 104-113. (DOI:10.1145/2466627.2466634)

Google Project Hosting (2011). *Blockly: A visual programming editor* (página web). Disponible en: https://code.google.com/p/blockly/ (consultado el 10 de enero de 2014).

Grover, S. (2009). "Computer science is not just for big kids", *Learning & Leading with Technology,* vol. 37, núm. 3, pp. 27-29.

Hug, S.; Guenther, R. y Wenk, M. (2013). "Cultivating a K12 computer science community: A case study", *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, Denver: ACM, pp. 275-280 (DOI:10.1145/2445196.2445278).

ICFES (2010). "Resultados del grado quinto en el área de matemáticas", en *Resultados históricos 2002, 2005, 2009*, Bogotá: Instituto Colombiano para la Evaluación de la Educación. Disponible en: http://www.icfessaber.edu.co/historico.php/graficar/nacion/id/1/grado/5/tipo/2 (consultado el 20 de enero de 2014).

INSA (2012). "Currículo INSA de Informática 2012", en *Eduteka* (página web), Cali: Instituto de Nuestra Señora de la Asunción. Disponible en: http://www.eduteka.org/tag/inicio/insa/1 (consultado el 15 de enero de 2014).

Javidi, G. y Sheybani, E. (2009). "Digispired: Digital inspiration for interactive game design and programming", *Journal of Computing Sciences in Colleges*, vol. 24, núm. 3, pp. 144-150.

Joyanes Aguilar, L. (2002). *Fundamentos de programación. Algoritmos, estructuras de datos y objetos,* Madrid: McGraw-Hill.

Kelleher, C. y Pausch, R. (2005). "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers", *ACM Computing Surveys*, vol. 37, núm. 2, pp.83-137 (doi:10.1145/1089733.1089734).

Kelleher, C.; Pausch, R. y Kiesler, S. (2007). "Storytelling Alice motivates middle school girls to learn computer programming", en *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* San José, California: ACM, pp. 1455-1464 (DOI:10.1145/1240624.1240844).

Kirriemuir, J. y McFarlane, A. (2004). *Futurelab series. Report 8: Literature review in games and learning,* Bristol: Futurelab.

Kliegman, R.; Stanton, B.; Schor, N.; St. Geme III, J. y Behrman, R. (2011). *Nelson textbook of pediatrics.* 19 ed., Filadelfia: Elsevier Saunders Inc.

Lameras, P.; Smith, D.; Moumoutzis, N.; Christodoulakis, S.; Ovcin, E. y Stylianakis, G. (2010). "Transforming teaching and learning: Changing the pedagogical approach to using educational programming languages", en *17th Association for Learning Technology Conference (ALT-C 2010),* Nottingham: ALT-C.

Lee, M. y Ko, A. (2011). "Personifying programming tool feedback improves novice programmers' learning", en *Proceedings of the Seventh International Workshop on Computing Education Research,* Providence: ACM, pp. 109-116 (DOI:10.1145/2016911.2016934).

Lin, C. C.; Zhang, M.; Beck, B. y Olsen, G. (2009). "Embedding computer science concepts in K-12 science curricula", en *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, Chattanooga: ACM, pp. 539-543 (DOI: 10.1145/1508865.1509050).

Liu, C. C.; Cheng, Y. B. y Huang, C. W. (2011). "The effect of simulation games on the learning of computational problem solving", *Computers & Education*, vol. 57, núm. 3, pp. 1907-1918. (DOI:10.1016/j.compedu.2011.04.002).

Meyers, A.; Cole; M.; Korth, E. y Pluta, S. (2009). "Musicomputation: Teaching computer science to teenage musicians", en *Proceedings of the Seventh ACM Conference on Creativity and Cognition*, Berkeley: ACM, pp. 29-38 (DOI:10.1145/1640233.1640241).

Ministerio de Educación Nacional (2006). *Estándares básicos de competencias de matemáticas.* Bogotá: Ministerio de Educación Nacional.

Ministerio de Educación Nacional (2008). *Ser competente en tecnología: ¡Una necesidad para el desarrollo! Orientaciones generales para la educación en tecnología,* Bogotá: Imprenta Nacional.

MIT Media Lab (2013). *Scratch Project* (página web). Disponible en: http://scratch.mit.edu/

Mitchell, A. y Savill-Smith, C. (2004). *The use of computer and video games for learning. A review of the literature,* Londres: Learning and Skills Development Agency.

National Academy of Sciences (2010). *Report of a workshop on the scope and nature of computational thinking,* Washington, DC: National Academies Press.

Osborne, J. y Hennessy, S. (2003). *Futurelab series. Report 6: Literature review in science education and the role of ICT: Promise, problems and future directions,* Bristol: Futurelab.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas,* Nueva York: Basic Books, Inc.

Repenning, A. y Ioannidou, A. (2008). "Broadening participation through scalable game design", *SIGCSE Bulletin*, vol. 40, núm. 1, pp. 305-309 (doi:10.1145/1352135.1352242).

Research Kitchen-Universiteit Van Amsterdam (2014). *RoboMind.net* (página web) Disponible en: http://www.robomind.net/es/index.html (consultado el 17 de agosto de 2014).

Resnick, M.; Maloney, J.; Monroy-Hernandez, A.; Rusk, N.; Eastmond, E.; Brennan, K.; Millner, A; Rosenbaum, E.; Silver, J.; Silverman, B. y Kafai, Y. (2009). "Scratch: Programming for all", *Communications of the ACM*, vol. 52, núm. 11, pp. 60-67 (DOI:10.1145/1592761.1592779).

Rizvi, M.; Humphries, T.; Major, D.; Jones, M. y Lauzun, H. (2011). "A CS0 course using Scratch", *Journal of Computing Sciences in Colleges*, vol. 26, núm. 3, pp. 19-27.

Rodger, S.; Bashford, M.; Dyck, L.; Hayes, J.; Liang, L.; Nelson, D. y Qin, H. (2010). "Enhancing K-12 education with Alice programming adventures", en *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, Bilkent: ACM, pp. 234-238 (DOI: 10.1145/1822090.1822156).

Rogozhkina, I. y Kushnirenko, A. (2011). PiktoMir: "Teaching programming concepts to preschoolers with anew tutorial environment", *World Conference on Educational Technology Research. Procedia - Social and Behavioral Sciences*, Moscú: Elsevier, pp. 601-605 (DOI: 10.1016/j.sbspro.2011.11.114).

Schwartz, J.; Stagner, J. y Morrison, W. (2006). "Kid's Programming Language (KPL)", en *ACM SIGGRAPH 2006. Educators program*, Boston: ACM, pp. 52.1-52.4 (DOI: 10.1145/1179295.1179348).

Sipitakiat, A. y Nusen, N. (2012). "Robo-Blocks: Designing debugging abilities in a tangible programming system for early primary school children", *Proceedings of the 11th International Conference on Interaction Design and Children*, Bremen: ACM, pp. 98-105 (DOI:10.1145/2307096.2307108).

Tarkan, S.; Sazawal, V.; Druin, A.; Golub, E.; Bonsignore, E.; Walsh, G. y Atrash, Z. (2010). "Toque: Designing a cooking-based programming language for and with children", en *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Atlanta: ACM, pp. 2417-2426 (DOI: 10.1145/1753326.1753692).

Tomcsányiová, M. y Tomcsányi, P. (2011). "Little beaver - A new bebras contest category for children aged 8-9", en *5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives, ISSEP 2011*, Bratislava: Springer-Verlag, pp. 201-212 (DOI: 10.1007/978-3-642-24722-4_18).

Utting, I.; Cooper, S.; Kölling, M.; Maloney, J. y Resnick, M. (2010). "Alice, Greenfoot, and Scratch - A discussion", *ACM Transactions on Computing Education*, vol. 10, núm.4, pp. 17.1-17.11 (DOI: 10.1145/1868358.1868364).

Valente, A. (2004). "Exploring theoretical computer science using paper toys (for kids)", en *ICALT '04 Proceedings of the IEEE International Conference on Advanced Learning Technologies,* Washington, DC: IEEE Computer Society, pp. 301-305 (DOI:10.1109/ICALT.2004.1357424).

Viewpoints Research Institute (2014). *Squeakland. Hogar de Squeak Etoys* (página web). Disponible en: http://www.squeakland.org/ (consultado el 19 de agosto de 2014).

Walton-Hadlock, M. (2008). "Tots to tweens: Age-appropriate technology programming for kids", *Children & Libraries: The Journal of the Association for Library*, vol. 6, núm. 3, p. 52-55.

Wen-Yu Lee, S. y Tsai, C. C. (2013). "Technology-supported learning in secondary and undergraduate biological education: Observations from literature review", *Journal of Science Education and Technology,* vol. 22, núm. 2, pp. 226-233 (doi:10.1007/s10956-012-9388-6).

Wolz, U.; Leitner, H.; Malan, D. y Maloney, J. (2009). "Starting with scratch in CS 1", en *SIGCSE'09 - Proceedings of the 40th ACM Technical Symposium on Computer Science Education,* Chattanooga: ACM, pp. 2-3 (doi:10.1145/1508865.1508869).

Woods, D. R. (2000). "An evidence-based strategy for problem solving", *Journal of Engineering Education*, vol. 89, núm. 4, pp. 443-459 (doi:10.1002/j.2168-9830.2000.tb00551.x).

Yaroslavski, D. (2013). *Lightbot: Programming puzzles* (página web). Disponible en: http://www.light-bot.com/ (consultado el 15 de enero de 2014).

Zuckerman, O.; Arida, S. y Resnick, M. (2005). "Extending tangible interfaces for education: Digital Montessori-inspired manipulatives", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Portland: ACM, pp. 859-868 (doi:10.1145/1054972.1055093).

Artículo recibido: 31 de julio de 2014
Dictaminado: 5 de noviembre de 2014
Segunda versión: 13 de enero de 2015
Aceptado: 26 de enero de 2015