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

CARLOS ANDRÉS PALMA SUAREZ

Estudiante de Maestría Grupo de Investigación de Preservación e Intercambio Digital de Información y Conocimiento - PRISMA de la Universidad Autónoma de Bucaramanga – UNAB. Avenida 42 No. 48 – 11 - Bucaramanga, Colombia. (57) 7 6436261. CE: cpalma@unab.edu.co. PALMA SUAREZ, C. A.

ROMAN EDUARDO SARMIENTO PORRAS

Docente Investigador Grupo de Investigación de Preservación e Intercambio Digital de Información y Conocimiento - PRISMA de la Universidad Autónoma de Bucaramanga – UNAB. Avenida 42 No. 48 – 11 - Bucaramanga, Colombia. (57) 7 6436261. CE: rsarmiento@unab.edu.co. SARMIENTO PORRAS, R. E.

Resumen:
Con el fin de establecer un modelo inicial de trabajo en el que logre enseñar la elaboración de macroinstrucciones a partir del entendimiento de procesos lógicos y matemáticos para mejorar las habilidades de resolución de problemas matemáticos en los estudiantes de 5° grado de educación básica primaria, se realizó una revisión del estado del arte en cuanto a experiencias significativas de enseñanza de esta temática, particularmente en lo referente a la comprensión y desarrollo de sus primeros algoritmos. Se determinaron importantes aspectos encontrados en estas experiencias, como las temáticas y contextos adecuados de enseñanza de la programación para niños, una caracterización de herramientas adecuadas para la programación en niños, una serie de técnicas de aplicación de la enseñanza de la programación en el aula y algunos ejemplos de modelos de evaluación de resultados en experiencias generadas de enseñanza con objeto de establecer una forma de evaluación para corroborar la efectividad del modelo de trabajo a proponer.

Abstract:
In order to establish an initial working model which achieves macro-instructions teaching from the understanding of logical and mathematical processes to improve the skills of mathematical problem solving in grade 5 students in basic primary education, a review of the state of the art in terms of meaningful experiences of teaching this subject was conducted, particularly with regard to the understanding and development of its first algorithms. Important aspects were found in these experiences like appropriate contexts and subjects teaching children's programming, a characterization of suitable tools for programming in children, a number of techniques for applying the teaching of programming in the classroom were determined and examples of assessment models generated results on learning experiences to establish a form of evaluation to corroborate the effectiveness of the working model to propose.

Palabras clave: Programación para Niños, Algoritmos, Experiencias, Estado del arte, Enseñanza, Competencias matemáticas.

Introducción

Durante los últimos años, como lo demuestran los resultados de las pruebas SABER de los años 2002 – 2005 – 2009 en el área de matemáticas en el nivel 5° de educación básica primaria (Instituto Colombiano para la Evaluación de la Educación - ICFES, 2010), se ha visto un deterioro notable en la calidad educativa en el área de matemáticas. Los últimos resultados son evidentes: revisando los niveles de desempeño en el área de matemáticas 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 del 50% de la población estudiantil, ni a nivel departamental ni a nivel nacional.

Una manera para contribuir a la solución de este gran problema no es encaminar únicamente a utilizar la tecnología para apoyar los procesos académicos, sino que también se ha buscado la enseñanza de la tecnología en sí, por lo que desde hace mucho tiempo se ha hecho hincapié en la enseñanza de la tecnología, particularmente de la enseñanza de la programación en los niños. Uno de los aspectos que abarca la enseñanza tecnológica está en la comprensión de los procesos lógicos como fases secuenciales de operaciones que transforman recursos con el fin de 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 se presenta en los Estándares Básicos de Competencias en Matemáticas (Ministerio de Educación Nacional, 2003). Según estos estándares, la formulación,

comparación y ejercitación de procedimientos, es un proceso en el cual 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

En el libro "Structure and Interpretation of Computer Programs", se habla sobre la importancia de esta enseñanza, 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-" (Abelson, et al., 1996). Iniciativas en otros países como los proyectos Code.org (2013), o Scratch entre muchos otros, buscan que mediante la programación, los niños aprendan a pensar creativamente, razonar sistemáticamente y trabajar colaborativamente (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 y colaboradores (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 es un tema que atañe, no solo al área de conocimiento tecnológico sino también al área matemática. Una propuesta de iniciar en los niños de 5° de primaria la enseñanza de elaboración de instrucciones secuenciales no suena ahora tan descabellada, sino más bien, se vuelve una herramienta complementaria 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 5° de educación básica primaria?.

Basados en la 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 (*palabras clave: teaching programming kids*). Se encontró mediante las librerías digitales SCOPUS, ACM Digital Library, ScienceDirect, IEEExplore y el motor de búsqueda Google Scholar un grupo de publicaciones entre artículos y sitios web, que a pesar de utilizar las palabras clave, muchas de ellas pertenecían a otras áreas de investigación, como psicología, medicina, entre otros, 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 documentos más citados, y a publicaciones más recientes con una ventana de observación un lapso de 10 años. Se ha abordado esta revisión de literatura bajo los siguientes temas:

- Experiencias de enseñanza de programación para niños y jóvenes
- Temáticas y contextos adecuado 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 que han brindado dichas experiencias, comprendiendo 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 tabla No. 1) que permitió identificar diferentes características de 18 experiencias de enseñanza, y con 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 (como por ejemplo Austria, Rep. Checa, Israel, Serbia, Dinamarca, Eslovaquia, Rusia), así como en Norteamérica (Estados Unidos, Canadá) y en Asia (Taiwan) se observa 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 niveles universitarios, implicando su aplicación en contextos temáticos diferentes de acuerdo a su nivel. Ninguna temática ha sido aplicada de manera forzada sino que se adaptaron al nivel tratado en el contexto de cada institución en cada 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 2 experiencias (Alt, et al., 2006), (Lin, et al., 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 y colaboradores (2006) utilizaron para ello el análisis de datos estadísticos para conocer el comportamiento de las redes sociales, mientras que Lin y colaboradores (2009) por su parte utilizaron el contexto de la bioinformatica y el análisis genético de cadenas de código ADN para la comprensión de relaciones evolutivas.

Tabla 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, USA	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, et al., 2011)	Taiwan	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, et al., 2009)	USA	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, et al., 2009)	USA	Entorno de desarrollo Scratch	Programación básica, programación avanzada (eventos, concurrencia, hilos); contexto diferente de acuerdo a 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
(Tomcsányiová & 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, et al., 2012)	USA	Plataforma GreenFoot	Conceptos básicos de programación, POO; contexto para programación de juegos	Jóvenes de grado 10° y 11°; ningún conocimiento sobre programación	Campamento de programación intensivo	Enseñanza de material instruccional preparado por estudiantes de niveles superiores
(Rogozhkina & Kushnirenko, 2011)	Rusia	Entorno PiktoMir	Conceptos fundamentales de programación; contexto sobre juego de control de robot para llenado de casillas	Preescolar (5 años y medio 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, pruebas aplicadas en computador

(Burke & Kafai, 2010)	USA	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 & Sheybani, 2009)	USA	Entornos de programación Kahootz y Squeak.	Programación para videojuegos. Contexto: temáticas sobre importancia de 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, autoreflexión.
(Kelleher, et al., 2007)	USA	Entorno de desarrollo Storytelling Alice	Programación básica. Contexto: creación de historias.	Estudiantes mujeres de educación básica y media (5° a 9°)	Enseñanza de herramienta y de conceptos de programación, quiz de conceptos y actitudes, construcción de historias.	Enseñanza a través de tutorial, uso de constructos de programación básica.
(Lee & Ko, 2011)	USA	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 de niveles académicos desde bachillerato hasta doctorados alrededor del mundo, sin ningún 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.
(Meyers, et al., 2009)	USA	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)	USA	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)	USA	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 & Nusen, 2012)	Tailandia	Sistema de construcción por bloques físicos Robo-Blocks	Programación para robotica. Contexto: creación de minirobots	Niños entre 8 y 9 años.	Creación de robots mediante construcción de bloques. Enseñanza de	Aprendizaje por construcción

					modelo de depuración.	
(Tarkan, et al., 2010)	USA	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, et al., 2005)	USA	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.

Fuente: Autor del proyecto

Temáticas y contextos para la enseñanza de la programación para niños

Se ha utilizado la enseñanza de la programación en muchos contextos, entre cuales se incluyen creación de historias (Burke & Kafai, 2010); (Kelleher, et al., 2007), conservación del medio ambiente y salud (Javidi & Sheybani, 2009), procesamiento de sonidos musicales (Meyers, et al., 2009), escenarios de cocina para preparación de recetas (Tarkan, et al., 2010), control de transporte (Liu, et al., 2011), así como también los clásicos contextos matemáticos como modelamiento (Abramovich, 2013); (Felleisen, et al., 2009), y tecnológicos, como programación de robots reales (Sipitakiat & Nusen, 2012), programación de computadores (Wolz, et al., 2009), uso de videojuegos para cumplimiento de objetivos (Lee & Ko, 2011); (Rodger, et al., 2010); (Rogozhkina & Kushnirenko, 2011); (Tomcsányiová & Tomcsányi, 2011), creación de videojuegos (Doerschuk, et al., 2012), flujos (Zuckerman, et al., 2005); (Valente, 2004), y multimedia (Rizvi, et al., 2011).

A pesar de todos estos diversos contextos y temáticas centrales, siempre se enseñó dentro de éstos 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 a muchas de las investigaciones encontradas, estos conjuntos de temas son lo que denominan constructos de programación. Revisando estos constructos 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 la explicar estos conceptos es fundamentalmente importante. Según Michael Kölling - citado en Utting y colaboradores (2010) – no se presenta un rango especial de edades para la enseñanza de la programación en general, incluso aseguró 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 que los niños de estas edades pudieron entender los conceptos de programación, no lograron lidiar con sintaxis complejas. En el experimento realizado por Sipitakiat y Nusen (2012), se intentó explicar las operaciones básicas de un nuevo entorno llamado Robo-Blocks a niños de 5 a 12 años, pero se dieron cuenta que los niños menores de 8 años a menudo tenían dificultad entendiendo estas operaciones, en particular por los comandos de girar, desplazarse, pues eran confusas las órdenes que se usaron. Estas razones permiten formalizar el rango ideal de edades en los estudiantes para poder iniciar esta investigación, el cual se establece en 10 años, siendo esta la edad promedio de los estudiantes que asisten a grado 5° 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 grado quinto 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 referentes a los constructos 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 al currículo revisado, estos temas se presentaron en este orden, encajando en los constructos recopilados en este informe. Dado el corto tiempo con el que se podría contar para la enseñanza de la programación para niños, no es posible cubrir todos estos temas en la metodología que se pretende plantear. Además, sería necesario revisar estos temas dependiendo también de la herramienta que se seleccione y que permita presentar los temas requeridos.

Por otra parte, se debe determinar un contexto apropiado para la aplicación de estos nuevos conceptos, enfocado principalmente a la resolución de problemas. Grover (2009) afirmó que la resolución de problemas, 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

estos temas es que vienen soportados sobre bases matemáticas, particularmente de niveles que se ven en los colegios. Basados en la revisión del estado del arte sobre las experiencias de aplicación, uno de los contextos encontrados con más frecuencia ha sido en la aplicación de la programación en el área de matemáticas. De acuerdo con los estándares de competencias para el área de matemáticas establecidos por el Ministerio de Educación Nacional (2003) para el grado 5°, los estudiantes de este nivel pueden construir diferentes tipos de modelos matemáticos y generar diversos métodos para la resolución de problemas referentes a 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

Justamente 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), el pensamiento algorítmico ayuda a los estudiantes a establecer una serie de pasos de un problema y plasmarlos en un programa, permitiendo que el estudiante pueda adquirir habilidades adicionales para definir y establecer claramente el problema, romper el problema 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 encontraron un sinnúmero de herramientas para la enseñanza de la programación para niños, cada uno con características muy diversas. Tarkan y colaboradores (2010), por ejemplo hablaron sobre una clasificación inicial de herramientas de programación para la enseñanza, dividiendo estas en dos grandes categorías. En una de ellas se habló sobre lenguajes de programación visuales, es decir, ambientes gráficos que se pueden usar para crear historias interactivas y juegos entre muchas otras. Dentro de estas se encuentran los lenguajes escritos mediante bloques de construcción (como por ejemplo Alice o Scratch), o mediante una sintaxis sencilla basada en reglas (como ToonTalk, Hands o Kodu). En la otra se encontraron los sistemas de programación tangibles, sistemas físicos para aprender a programar, tales como bloques físicos (Zuckerman, et al., 2005); (Sipitakiat & Nusen, 2012), o incluso cartas físicas (Valente, 2004).

Kelleher y Pausch (2005) propusieron una especie de clasificación taxonómica de estas herramientas. Lo interesante de este estudio ha sido la clasificación en cuanto a diversos atributos con los que cuenta cada herramienta de acuerdo a los criterios de enseñanza que se deben manejar. Estos atributos fueron los siguientes:

- Estilo de programación (procedimental –estructural-, funcional, orientado a objetos, basado en objetos, orientado a eventos, basado en eventos y basado 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, removiendo 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 y colaboradores (2010) generaron otra propuesta de caracterización de herramientas 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 IDE (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 habló sobre la necesidad de trabajar con lenguajes de programación especialmente diseñados para la enseñanza de la programación. Como un ejemplo, Schwartz, Stagner y Morrison (2006) propusieron un lenguaje de programación especializado para niños llamado KPL, y propusieron criterios especiales llamados puntos de diseño que tuvieron en cuenta para la creación de KPL. Entre ellos se encuentran la diversión que generan, 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 permitirle conocer otro lenguaje de programación, la consistencia 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 lo es la sencillez de la sintaxis, por encima de la importancia de un ambiente gráfico. Muchos críticos aseguraron 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 entendían cómo resolver problemas usando estructuras de programas puesto que se estancaban en errores sintácticos, y no se veían claramente los beneficios educativos (Sipitakiat & Nusen, 2012). Según afirmaron 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 lo que hace es perjudicar más que ayudar. De acuerdo con Stephen Cooper – citado en Utting y colaboradores (2010) – gran parte de las primeras frustraciones asociadas con la depuración (solución de errores) se minimizan utilizando un entornos fácil de trabajar, pues así que los estudiantes son capaces de avanzar por su cuenta, en su intento de resolver problemas específicos. Fidge y Teague (2009) hablaron sobre la programación "libre de sintaxis" con el fin de separar la noción de codificación de la habilidad para resolver problemas mediante programación.

Toda esta revisión de investigaciones previas ha arrojado como resultado varias características importantes que debería tener la herramienta o herramientas a ser utilizadas en el curso:

- Debe soportar los primeros constructos 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) sean en español.

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 Jean Piaget (Kliegman, et al., 2011). De acuerdo a esta teoría, Piaget 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 la 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 teorías del aprendizaje, como el construccionismo, propuesto por Seymour Papert. Según 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 en los modelos que tiene disponibles en su mente.

La edad apropiada para poder aprender estos nuevos conceptos y que 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 fáciles de utilizar y que puedan fácilmente ser enlazados con otros tipos de medios, como libros o tutoriales. El nuevo modelo que se pretende enseñar en este proyecto está basado en el creación de un pensamiento algorítmico, con el fin 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 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 debiera tenerla en esta época actual.

Para lograr transmitir esta habilidad en forma eficaz a los estudiantes, sería necesario entonces generar un curso bien estructurado y diseñado. Con el fin de poder generar un buen diseño de un curso que se fundamente en la enseñanza de la programación, se debe enfocar particularmente a desarrollar las habilidades de los estudiantes por aplicar su conocimiento a la solución de problemas reales mientras se logre al mismo tiempo generar programas coherentemente desarrollados (Lameras, et al., 2010). Resnick y colaboradores (2009) hablan particularmente sobre 3 habilidades que se adquieren en la enseñanza de la programación: habilidad para pensar creativamente, habilidad para razonar sistemáticamente y habilidad para trabajar colaborativamente. Según Resnick, 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 & 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 al respecto del beneficio del uso de los videojuegos en la educación, así como también se encontró amplia información sobre la no conveniencia de estos tipos de elementos en el ámbito educativo. Mitchell y Savill-Smith en su estado del arte (2004), 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, encontraron también que el uso frecuente de juegos de computador, especialmente los que contengan algún contenido violento, que 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 utilizar estos tipos de herramientas es la adecuación de éstos al contexto actual de trabajo en el grupo. Malone – citado en Kirriemuir y McFarlane (2004, p. 4) – indicó que para que se logre el uso efectivo de un juego contextualizado en una experiencia de aprendizaje, las actividades que se les propongan 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 no caen en 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 objetivo, o incluso que deban recurrir a técnicas como el ensayo y error para el desarrollo de su proyecto. Según Stephen Cooper – citado en Utting y colaboradores (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 la resolución de problemas reales en diferentes contextos, preparados con cuidado de acuerdo a la temática a trabajar en cada sesión y pudiendo detectar algunas de las múltiples posibilidades de solución que se generaran en el desarrollo del proyecto, durante la enseñanza de los nuevos modelos algorítmicos que se presenten.

Igualmente, existen otras formas adicionales que han sido aplicadas exitosamente en la enseñanza por medio de tecnologías de la información, no solo aplicados en la temática de la programación, sino también en otros contextos. Por ejemplo, Wen-Yu Lee y Tsai (2013) en su revisión de literatura, encontraron adicionalmente varios tipos de técnicas de enseñanza que implicaban en uso de tecnologías de la computación, y especialmente para fomentar habilidades de orden superior como las habilidades de resolución de problemas o incluso el pensamiento crítico, tales como el uso de simulaciones, los materiales multimedia, los sistemas integrados, cursos mixtos, los tutoriales y las evaluaciones asistidas por computador, entre otros. Sin embargo, estas técnicas de enseñanza requerirían esfuerzos interdisciplinarios mayores y tiempos mayores de preparación curricular que el que podría generar el desarrollo de proyecto o el uso de un juego existente. Uno de los aspectos para tener en cuenta para la aplicación de este tipo de proyectos que integren la enseñanza de la programación en un entorno académico es la revisión de restricciones que se pudieran presentar, ya que se debe buscar no solo que hayan 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 tanto para el docente que podría impartir su clase, como para el estudiante que reciba este conocimiento. 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 lograr 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.

Modelos de evaluación

En los experimentos de Agina (2012), se tomaron en cuenta una serie de variables en la realización de sus pruebas, como la cantidad de ejercicios ejecutados, el tiempo que le tomó desarrollar cada

ejercicio, el número de respuestas correctas e incorrectas y el número de tareas que no pudo completar, entre otros. En ese mismo experimento se tomó una variable adicional para la medición del pensamiento creativo en la resolución de problemas, como la necesidad de los estudiantes en la solicitud de una ayuda extra por parte del 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 las pruebas 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 cual se realizaron preguntas para determinar el grado de satisfacción, utilidad y facilidad de las pruebas. Estas variables ayudaron a determinar si han adquirido intrínsecamente la habilidad para comprender los problemas que se le 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 un mejor aprendizaje y una mejor forma de resolver problemas. Este mismo estado se nombró en 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 de flujo basado en el modelo de flujo de 3 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 una serie de preguntas de tipo matemático para resolver problemas referentes a una temática recientemente vista, y preguntas adicionales sobre las opiniones que tiene sobre la prueba. Esta prueba debería estar basada en lo que normalmente ven los estudiantes en el área de matemáticas, prácticamente de la misma forma a una prueba tradicional que desarrollaría el docente en su área. La prueba 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 vio en el experimento de Giannakos y Jaccheri (2013), en el cual se realizaban preguntas para determinar el grado de satisfacción, utilidad y facilidad de las pruebas. Esta variable ayudaría en parte a determinar si se ha adquirido intrínsecamente la habilidad para comprender los problemas que se le presentaron en la prueba, además de servir como 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 de esta temática en lo referente a la comprensión y desarrollo de sus primeros algoritmos se lograron determinar importantes aspectos encontrados en estas experiencias. En cuanto a las temáticas y contextos adecuados de enseñanza de la programación para niños, se logra determinar una serie de tópicos en las áreas de matemáticas, estadística y geometría que pueden ser abordados mediante aplicaciones prácticas algorítmicas, utilizando para ello varios constructos de programación a nivel básico, como el uso de variables y diversas estructuras de control. En referencia a las características

de herramientas adecuadas para la programación en niños, aunque es obviamente necesario que las herramientas puedan cubrir los constructos que se pretenden enseñar, es también necesario que se cuente 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 error y la posible forma de generar una solución adecuada.

Con respecto a las técnicas de aplicación de la enseñanza de la programación en el aula, se observa que la generación de proyectos y el uso de los juegos, entre otras técnicas, son adecuadas para brindar una experiencia de enseñanza de la programación. A su vez, se logró determinar un modelo de evaluación de resultados que permita medir el mejoramiento de las competencias de resolución de problemas matemáticos y así establecer la efectividad del modelo de trabajo 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 y poder ser implantado en una institución académica, y poder medir su efectividad en el mejoramiento de las habilidades matemáticas en los niños.

Bibliografía

Abelson, H., Sussman, G. J. & 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*, Enero, 30(1-2), pp. 4-11.

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*, Julio, 28(4), pp. 1213-1226.

Alonso, S., Cordón, O., Fernández de Viana, I. & Herrera, F., 2003. *Análisis de distintas vertientes para la paralelización de los algoritmos de Optimización basada en Colonias de Hormigas*. Gijón, Universidad de Granada, pp. 160-167.

Alt, C. y otros, 2006. *Social Networks Generate Interest in Computer Science*. New York, ACM, pp. 438-442.

Arias Galeano, E. A. & Cáceres Tabares, R. F., 2012. *Guías con componente informático para el aprendizaje de los principios de inteligencia de enjambres*, Bucaramanga: UDI.

Arias, M., López, A. & Rosario, H., 2002. *Metodología Dinámica para el Desarrollo de Software Educativo*. Valencia, Virtual Educa 2002.

Burke, Q. & Kafai, Y. B., 2010. *Programming & Storytelling: Opportunities for Learning About Coding & Composition*. Barcelona, ACM, pp. 348-351.

Casallas Fonseca, C. E. & Vargas Aristizábal, S. F., 2010. Bioescenarios IV Inteligencia de enjambres: Optimizacion por cúmulo de particulas. *Sistemas- Acis*, Volumen 113.

Cataldi, Z., Lage, F., Pessacq, R. & García-Martínez, R., 2003. Metodología Extendida Para La Creación De Software Educativo Desde Una Visión Integradora. *Revista Latinoamericana de Tecnología Educativa - RELATEC, vol 2, no. 1.*.

Chen, P.-H., 2009. Particle Swarm Optimization for Power Dispatch with Pumped Hydro. En: A. Lazinica, ed. *Particle swarm optimization.* s.l.:InTech, p. 476.

Code.org, 2013. *Code.org.* [En línea] Available at: http://code.org/

Delgado, J. y otros, 2013. Aprendizaje de la programación en el Citilab. *Revista Iberoamericana de Ciencia, Tecnología y Sociedad,* pp. 123-133.

Doerschuk, P., Liu, J. & Mann, J., 2012. *An INSPIRED game programming academy for high school students.* s.l., s.n.

Dorigo, M. & Gambardella, L. M., 1996. *Ant colonies for the traveling salesman problem,* Bruselas: Université Libre de Bruxelles.

Dorigo, M. & Gambardella, L. M., 1996. *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem,* Bruselas: Université Libre de Bruxelles.

Felleisen, M., Findler, R., Flatt, M. & Krishnamurthi, S., 2009. *A functional I/O system*: Or, fun for freshman kids.* s.l., ICFP, pp. 47-58.

Fidge, C. & Teague, D., 2009. *Losing Their Marbles: Syntax-free Programming for Assessing Problem-solving Skills.* Darlinghurst, Australian Computer Society, Inc., pp. 75-82.

Floreano, D. & Mattiussi, C., 2012. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies.* Cambridge - London: MIT Press.

Galvis Panqueva, Á. H., 2001. *Ingeniería de Software Educativo. 3a reimpresión de la 1a edición.*. Bogotá: Uniandes.

Galvis Panqueva, Á. H., Mariño Drews, O. & Gómez Castro, R. A., 1998. *Ingeniería de software educativo con modelaje orientado por objetos: un medio para desarrollar Micromundos interactivos,* Bogotá: Universidad de los Andes.

Gao, S. & Cao, C., 2012. Convergence Analysis of Particle Swarm Optimization Algorithm. *Advances in information Sciences and Service Sciences(AISS),* 4(14).

Giannakos, M. & Jaccheri, L., 2013. *What Motivates Children to Become Creators of Digital Enriched Artifacts?.* New York, ACM, pp. 104-113.

Grover, S., 2009. Computer Science Is Not Just for Big Kids. *Learning & Leading with Technology,* Noviembre, 37(3), pp. 27-29.

Hartmanis, J. & Stearns, R., 1965. On te computational complexity of algorithms. *Transactions of the American Mathematical Society,* Volumen 117, pp. 285--306.

Hernández Valdelamar, E. J., 2005. *Diseño Instruccional Aplicado Al Desarrollo De Software Educativo.* Mexico D.F.: Fundación Arturo Rosenblueth.

Hug, S., Guenther, R. & Wenk, M., 2013. *Cultivating a K12 Computer Science Community: A Case Study.* New York, ACM, pp. 275-280.

Instituto Colombiano para la Evaluación de la Educación - ICFES, 2010. *Resultados del grado Quinto en el área de Matemáticas.* [En línea]
Available at: http://www.icfessaber.edu.co/historico.php/graficar/nacion/id/1/grado/5/tipo/2

Instituto de Nuestra Señora de la Asunción - INSA, 2012. *Currículo INSA de Informática 2012.* [En línea]
Available at: http://www.eduteka.org/tag/inicio/insa/1

Javidi, G. & Sheybani, E., 2009. Digispired: Digital Inspiration for Interactive Game Design and Programming. *Journal of Computing Sciences in Colleges,* Enero, 24(3), pp. 144-150.

Jimenez B., D. F. & Peñuela, D. F., 2007. Escenarios para el aprendizaje de tendencias bioinspiradas. *Sistemas ACIS.*

Kelleher, C. & Pausch, R., 2005. Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys,* Junio, 37(2), pp. 83-137.

Kelleher, C., Pausch, R. & Kiesler, S., 2007. *Storytelling Alice Motivates Middle School Girls to Learn Computer Programming.* New York, ACM, pp. 1455-1464.

Kennedy, J. & Eberhart, R. C., 1995. *Particle swarm optimization.* Piscataway, IEEE.

Kirriemuir, J. & McFarlane, A., 2004. *Futurelab Series. Report 8: Literature Review in Games and Learning,* Bristol: Futurelab.

Kliegman, R. y otros, 2011. *Nelson Textbook of Pediatrics.* Philadelphia: Elsevier Saunders Inc..

Lameras, P. y otros, 2010. *Transforming teaching and learning: Changing the pedagogical approach to using educational programming languages.* Nottingham, ALT-C.

Lee, M. & Ko, A., 2011. *Personifying Programming Tool Feedback Improves Novice Programmers' Learning.* New York, ACM, pp. 109-116.

Lin, C.-C., Zhang, M., Beck, B. & Olsen, G., 2009. *Embedding Computer Science Concepts in K-12 Science Curricula.* New York, ACM, pp. 539-543.

Liu, C.-C., Cheng, Y.-B. & Huang, C.-W., 2011. The effect of simulation games on the learning of computational problem solving. *Computers & Education,* Noviembre, 57(3), pp. 1907-1918.

Marquès Graels, P., 2002. *Diseño y Desarrollo Multimedia.* [En línea]
Available at: http://www.peremarques.net/disdesa.htm

Meyers, A., Cole, M., Korth, E. & Pluta, S., 2009. *Musicomputation: Teaching Computer Science to Teenage Musicians.* New York, ACM, pp. 29-38.

Ministerio de Educación Nacional, 2003. *Estándares Básicos de Competencias de Matemáticas,* Bogotá: s.n.

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

Ministerio de Educación Nacional, 2012. *Colombia Aprende.* [En línea]
Available at: http://www.colombiaaprende.edu.co/

MIT Media Lab, 2013. *Scratch Project.* [En línea] Available at: http://scratch.mit.edu/

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

Muñoz, M. A., Lopez, J. A. & Caicedo, E., 2008. Swarm intelligence: problem-solving societies (a review). *Revista Ingeniería e Investigación,* 28(2), pp. 119-130.

National Academy of Sciences, 2010. *Report of a Workshop on The Scope and Nature of Computational Thinking,* Washington, D.C.: National Academies Press.

Osborne, J. & Hennessy, S., 2003. *Futurelab Series. Report 6: Literature Review in Science Education and the Role of ICT: Promise, Problems and Future Directions,* Bristol: Futurelab.

Palma Suárez, C. A., Díaz Ribero, S. & Lizcano Dallos, A. R., 2011. *Metodología híbrida para reingeniería y desarrollo de software educativo.* Bucaramanga, s.n.

Papert, S., 1980. *Mindstorms: Children, Computers, and Powerful Ideas.* New York: Basic Books, Inc..

Peláez Camarena, G. & López Azamar, B., 2006. Metodología para el Desarrollo de Software Educativo (DESED). *UPIICSA,* Mayo-Diciembre.Issue 41-42.

Poli, R., Kennedy, J. & Blackwell, T., 2007. Particle swarm optimization - An overview. *Swarm Intell,* Issue DOI 10.1007/s11721-007-0002-0, pp. 33-57.

Ramírez M., A. y otros, 1999. *Software Educativo: Metodología de desarrollo e incorporación en los ambientes de aprendizaje,* Bogotá: Universidad EAFIT y Universidad Pontificia Bolivariana.

Repenning, A. & Ioannidou, A., 2008. Broadening Participation through Scalable Game Design. *SIGCSE Bulletin,* Marzo, 40(1), pp. 305-309.

Resnick, M. y otros, 2009. Scratch: Programming for All. *Communications of the ACM,* Noviembre, 52(11), pp. 60-67.

Rizvi, M. y otros, 2011. A CS0 Course Using Scratch. *Journal of Computing Sciences in Colleges,* Enero, 26(3), pp. 19-27.

Rodger, S. y otros, 2010. *Enhancing K-12 Education with Alice Programming Adventures.* New York, ACM, pp. 234-238.

Rodríguez, S. M. & Mora M, J. J., 2008. Bioescenarios III: Sistemas inmunológicos artificiales. *SISTEMAS -ACIS,* Volumen 107.

Rogozhkina, I. & Kushnirenko, A., 2011. *PiktoMir: Teaching programming concepts to preschoolers with anew tutorial environment.* Moscu, s.n.

Ruffini, M. F., 2000. Do It Step-by-Step: A Systematic Approach to Designing Multimedia Projects. *Learning & Leading with Technology. International Society for Technology in Education ISTE.*

Schutte, J. F. y otros, 2004. Parallel global optimization with the particle swarm algorithm. *International Journal for Numerical Methods in Engineering,* 61(13), pp. 2296-2315.

Schwartz, J., Stagner, J. & Morrison, W., 2006. *Kid's Programming Language (KPL).* Boston, ACM, pp. 52.1-52.4.

Shayegui, A., Shayegui, H. & Eimani Kalasar, H., 2009. Application of PSO Technique for Seismic Control of Tall Building. *International Journal of Electrical and Computer Engineering*, 4(5), pp. 293-300.

Sipitakiat, A. & Nusen, N., 2012. *Robo-Blocks: Designing Debugging Abilities in a Tangible Programming System for Early Primary School Children.* New York, ACM, pp. 98-105.

Tarkan, S. y otros, 2010. *Toque: Designing a Cooking-based Programming Language for and with Children.* New York, ACM, pp. 2417-2426.

Tomcsányiová, M. & Tomcsányi, P., 2011. *Little beaver - A new bebras contest category for children aged 8-9.* Bratislava, ISSEP.

Umarani, R. & Selvi, V., 2010. Particle Swarm Optimization - Evolution, Overview and Applications. *International Journal of Engineering Science and Technology,* 2(7), pp. 2802-2806.

Utting, I. y otros, 2010. Alice, Greenfoot, and Scratch - A Discussion. *ACM Transactions on Computing Education,* Noviembre, 10(4), pp. 17.1 - 17.11.

Valencia, M. E., 2003. *Un Método de Desarrollo de Aplicaciones Educativas Hipermedia.* [En línea] Available at: http://eisc.univalle.edu.co/materias/multimedia/material/metdesaplichm.pdf

Valente, A., 2004. *Exploring theoretical computer science using paper toys (for kids).* Washington, s.n.

Walton-Hadlock, M., 2008. Tots to Tweens: Age-Appropriate Technology Programming for Kids. *Children & Libraries: The Journal of the Association for Library*, 6(3), pp. 52-55.

Wen-Yu Lee, S. & Tsai, C.-C., 2013. Technology-supported Learning in Secondary and Undergraduate Biological Education: Observations from Literature Review. *Journal of Science Education and Technology*, 22(2), pp. 226-233.

Wolz, U., Leitner, H., Malan, D. & Maloney, J., 2009. *Starting with scratch in CS 1.* s.l., s.n.

Ying, T., Shi, Y., Shai, Y. & Wan, G., 2011. *Advances in Swarm Intelligence, Part I.* s.l.:Springer.

Zuckerman, O., Arida, S. & Resnick, M., 2005. *Extending Tangible Interfaces for Education: Digital Montessori-inspired Manipulatives.* New York, ACM, pp. 859-868.