PROMOVER HABILIDADES DE PENSAMIENTO COMPUTACIONAL EN LA ERA DE LA INEQUIDAD TECNOLÓGICA

Erika Daza-Pérez*
https://orcid.org/0000-0002-7549-9198
erdaza1212@yahoo.es

Fanny L. Lizarazo*
https://orcid.org/0000-0001-7165-5141
fannylicapi@hotmail.com

*Colegio Integrado Divino Niño, Capitanejo
Santander, Colombia

Cita este capítulo:

Daza-Pérez, E. y Lizarazo, F. L. (2020). Promover habilidades de pensamiento computacional en la era de la inequidad tecnológica. En: Villota Enríquez, J. A. y González Valencia, H. *Tecnología, Sociedad y Educación: perspectivas interdisciplinares en torno a las TIC desde el campo social y educativo* (pp. 127-143). Cali, Colombia: Editorial Universidad Santiago de Cali.

Promover habilidades de pensamiento computacional en la era de la inequidad tecnológica

Erika Daza-Pérez
Fanny L. Lizarazo

Resumen

Este documento presenta resultados parciales de un proyecto curricular y de investigación que busca promover habilidades de pensamiento computacional mediante la programación con Scratch en educación básica. Se centra en uno de los objetivos específicos del proyecto que es identificar habilidades de pensamiento computacional en estudiantes de una institución educativa oficial de Colombia. Se aplicó el test de pensamiento computacional propuesto por Román-González (2016) a 224 estudiantes que cursaban los grados séptimo, octavo y noveno de básica secundaria para evaluar, mediante 28 ítems, conceptos y tareas de pensamiento computacional. Según los resultados, el 52.51% de los estudiantes, logró resolver de forma adecuada la mayoría de los ítems propuestos en el cuestionario. Los ítems del concepto direcciones fueron desarrollados correctamente por el 77.5% mientras que en condicional simple, los estudiantes tuvieron menor porcentaje de respuestas correctas, solo el 23%. En la tarea secuenciación el 52% de estudiantes tuvo dificultades, mientras que depurar fue la tarea donde el grupo en general alcanzó mejor desempeño. El perfil de habilidades, a nivel de conceptos y tareas indica que se debe profundizar en el análisis de los resultados del test así como de los prototipos de videojuegos creados por los estudiantes. Apoyado en ello, la reflexión a nivel de los procesos de aula pone en evidencia la necesidad de ajustar las acciones didácticas y curriculares y fortalecer la formación de docentes.

Palabras clave

Pensamiento computacional, Scratch, programación, enseñanza de la tecnología, enseñanza de la informática.

Abstract

This document presents partial results of a research and curricular project aims at promoting computational thinking skills by programming with Scratch in a secondary school. Research focuses on one of the specific objectives of the project, which is to identify students computational thinking skills within an official educational institution in Colombia. The Computational Thinking Test (CTr) proposed by Román-González (2016) was applied to 224 students in seventh, eighth and ninth grades in order to assess computational thinking concepts and tasks through 28 items. According to the results, 52.51% of the students were able to adequately solve most of the items proposed in the questionnaire. The items of the concept "directions" were developed correctly by 77.5% while in "simple conditional", students had lower percentage of correct answers, only 23%. In the "sequencing" task, 52% of students had difficulties, while debugging was the task where the group in general achieved better performance. Skills profile, in terms of concepts and tasks, indicates that the analysis of the test results as well as the video game prototypes created by the students should be deepened. Based on this, educational processes in the classroom proves the need to adjust teaching styles and curricular actions and strengthen teacher training.

Key words
Computational thinking, Scratch, programming, teaching technology, teaching computer science.

Introducción

Las acciones orientadas al fortalecimiento de las habilidades involucradas en el pensamiento computacional y la capacidad para usar los recursos digitales de forma eficaz enfrentan dificultades y cierto atraso, principalmente en Latinoamérica. Esta situación está asociada con las concepciones limitadas que sustentan diversas políticas y reformas (Vázquez, Bottamedi y Brizuela, 2019), con la infraestructura disponible, formación de profesores (Buss y Gamboa, 2017; Reding et al., 2016) y los procesos de enseñanza y aprendizaje que se adelantan en las instituciones públicas a nivel de educación básica.

Particularmente en Colombia, las orientaciones curriculares oficiales han sido propuestas para el área de tecnología (MEN, 2008) y no hacen referencia explícita al desarrollo habilidades como la programación, la representación de datos a través de abstracciones o el pensamiento algorítmico. En relación con los programas de formación de profesores, iniciativas focalizadas al aprendizaje de la programación son escasas y están en etapa inicial, por ejemplo: Coding for Kids[20] y los espacios

20 https://www.eltiempo.com/tecnosfera/novedades-tecnologia/coding-for-kids-el-plan-que-ensenara-a-programar-en-colegios-publicos-354574.

centrados en gamificación y creación de videojuegos que fueron dispuestos en el pasado encuentro nacional del programa Computadores para Educar[21].

Bajo estas consideraciones, reconociendo el pensamiento computacional y con él, la programación como procesos fundamentales en la educación básica, desde el año 2016, se desarrolla un proyecto cuyo objetivo principal es promover habilidades de pensamiento computacional en estudiantes desde los cinco hasta los dicisiete años (preescolar hasta finalizar la secundaria) teniendo como base el aprendizaje de la programación con Scratch. Con ello se pretende motivar la cualificación de algunos docentes y establecer el pensamiento computacional como eje de la planeación curricular en el área de tecnología e informática desde la interdisciplinariedad e integralidad de los procesos didácticos.

El punto de partida atendió necesidades particulares de infraestructura, conectividad, problemas de motivación y aprendizaje en un contexto rural (Daza-Pérez y Santoyo 2016). Posteriormente se introdujo un grupo de profesores de primaria y la profesora de informática de una institución educativa oficial en el aprendizaje autónomo de la programación con Scratch quienes la integraron en la planeación curricular y abordaron en el área de tecnología e informática.

De esta forma se consolida una propuesta curricular y de investigación que pretende promover el pensamiento computacional y con ello, la interdisciplinariedad e integralidad en los procesos de enseñanza que forman parte de la propuesta pedagógica de la institución educativa orientada.

Desde el 2017 se han introducido algunos principios básicos de programación con Scratch en la elaboración de videojuegos y analizado componentes del pensamiento computacional en estudiantes de grado séptimo, octavo y noveno de básica secundaria (12 – 17 años), resultados que constituyen el elemento central de este artículo y que permiten generar elementos para revisar las acciones de aula, identificar componentes del pensamiento computacional que deben ser fortalecidos y aquellos que deben ser integrados al currículo. También para formular mecanismos a seguir en la institucionalización de los estándares de pensamiento computacional y demás acciones que permitirán desarrollar la segunda fase de la propuesta curricular y de investigación. Se presentan entonces, resultados sobre conceptos y tareas computacionales identificadas en los grupos de estudiantes citados.

21 http://www.computadoresparaeducar.gov.co/educadigital2019/talleres.php

Referente teórico

Aprender a pensar computacionalmente es reconocida como una acción fundamental inherente al desarrollo tecnológico, donde la programación y la computación sustentan la comunicación, la ciencia, la cultura y los negocios en nuestra sociedad digital (Howland y Good, 2015). Este proceso, visto como un conjunto de habilidades esenciales que también están involucradas en el aprendizaje de la matemática, el lenguaje, las ciencias naturales, posibilita crear, pensar sobre la tecnología y sus principios básicos de funcionamiento en vez de sólo consumirla (Resnick et al., 2009, Zapata-Ros, 2015).

Pensamiento computacional (PC) es un término propuesto por Wing (2006) para referirse a las habilidades de pensamiento, hábitos y enfoques involucrados en la resolución de problemas apoyados en un computador. Lo definió como "un proceso que envuelve resolver problemas diseñar sistemas y entender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática". Implica definir, comprender y resolver problemas; así como el razonamiento en múltiples niveles de abstracción, la comprensión, aplicación, automatización y análisis de la idoneidad de las abstracciones.

Este proceso admite así varias definiciones dada la diversidad de acciones, actividades y sub procesos que lo constituyen. También porque integra varios tipos de pensamiento (pensamiento algorítmico, pensamiento de ingeniería, pensamiento de diseño y pensamiento matemático) (Lee y Malyn-Smith, 2019) y diversos elementos fundamentales en el aprendizaje, comunes con otras áreas.

Desde el proyecto 'CAS Barefoot' (CAS, 2015) se reconocen en él seis conceptos ('lógica', 'algoritmos', 'descomposición', 'patrones', 'abstracción', y 'evaluación sistemática') y cinco aproximaciones ('experimentación', 'creación', 'depuración', 'perseverancia', y 'colaboración), mientras que Brennan y Resnick (2012) con el modelo MIT-Harvard lo abordan desde tres dimensiones que se derivan del análisis sobre las experiencias con Scratch: conceptos, prácticas y perspectivas computacionales.

a. Conceptos: corresponden a lo que aprenden los estudiantes y comprende:

- Secuencias: implica identificar y expresar una actividad o tarea como una serie de pasos individuales (discretos) y ordenados, que puedan ser ejecutados por un ordenador.
- Bucles: es una instrucción que se repite hasta que se cumpla cierta condición.

Eventos: cuando algo pasa, entonces causa que otra cosa ocurra. Paralelismos: se refiere a varias secuencias de instrucciones que se ejecutan al mismo tiempo, simultáneamente ('en paralelo').

- Condicionales: la capacidad de tomar decisiones basadas en ciertos estados o situaciones.
- Operadores: permiten al programador incluir expresiones lógicas, matemáticas y de cadena, en sus programas.
- Datos: incluye el almacenamiento, recuperación y actualización de valores en un programa.

b. Prácticas computacionales ¿Cómo lo aprenden?: se refieren a qué tipo de procesos y prácticas ponen en marcha los niños cuando construyen sus programas. Comprende: experimentación e iteración, evaluación y depuración, reutilización y remezcla y abstracción y modularización.

c. Perspectivas computacionales, ¿Para qué lo aprenden?: implica: expresarse, conectarse, interrogarse.

Las dimensiones descritas están involucradas en prácticas de programación apoyadas en Scratch, creado por Lifelong Kindergarten Group en el MIT Media Laboratory. Scratch es un entorno de programación visual que permite a los usuarios crear proyectos multimedia interactivos. La programación se realiza ensamblando bloques de comandos, de diferentes colores, para controlar objetos gráficos en 2-D que se mueven en un fondo llamado "escenario" (stage) (Maloney et al., 2010).

Esta aplicación, basada en ideas constructivistas de Logo permite al estudiante experimentar con los objetos o con herramientas para crear sus propias estrategias en las que enfrentará diversas situaciones que implican resolver problemas. Con ello, se ofrecen elementos que facilitan a los estudiantes a comprender la programación al ofrecer una perspectiva diferente de entornos más tradicionales.

Por su naturaleza visual y un método intuitivo de programación de arrastrar y soltar, Scratch es ideal para estudiantes de secundaria con poca experiencia en programación (Sáez-López et al. 2016). Además de trabajar en historias interactivas, juegos y animaciones individualmente o en colaboración con sus compañeros, los estudiantes pueden aprender conceptos matemáticos y computacionales (Maloney et al., 2010; Resnick et al., 2009).

En consecuencia con lo descrito, el objetivo de investigación que se aborda en el presente artículo es identificar habilidades de pensamiento computacional; se

sustenta en las dimensiones conceptos y prácticas computacionales (tareas) consideradas en el modelo MIT-Harvard. Por tanto, la metodología así como los resultados y el análisis correspondiente se centran en estas dos dimensiones del pensamiento computacional.

Metodología

Se trata de un proyecto con fines curriculares, que se apoya en la investigación. El desarrollo del componente curricular comprendió: una etapa de motivación, una etapa de capacitación, una etapa de integración en la planeación y la etapa de implementación que estuvo estrechamente ligada con el componente investigativo.

En la investigación, la etapa inicial correspondió a la identificación de habilidades de pensamiento computacional. Para ello se aplicó el test de Román-González (2016) que consta de 28 ítems y se evaluaron tres tares cognitivas: secuenciación, completamiento y depuración, asociadas con siete conceptos computacionales que también fueron evaluados y están ordenados en una dificultad creciente (cuatro ítems para cada caso): 1) *direcciones o secuencias básicas*, 2) *bucles – 'repetir veces'*, 3) *bucles – 'repetir hasta'*, 4) *condicional simple – 'if',* 5) *condicional compuesto – 'if/ else', mientras que* – 6) *'while' y funciones simples.*

El test fue aplicado en las clases de informática, a estudiantes que habían recibido desde el inicio de la secundaria, orientación y desarrollado ejercicios básicos de programación en Scratch apoyados en el cuaderno de trabajo propuesto por López (2011). De esta manera, en el grado séptimo se implementaron actividades tales como: movimientos (ubicación en el plano cartesiano), diálogos, manejo de tiempos, crear objetos. En los grados octavo y noveno los mismos ejercicios anteriores junto con manejo de variables: sumar puntos, restar, ganar o perder y cambios de escenarios.

Durante la etapa investigativa participaron 224 estudiantes de los cuales 117 eran mujeres y 110 hombres. Por cada grado estuvieron distribuidos así:

- 71 estudiantes de séptimo. Es decir, segundo año de educación básica secundaria con edades entre los 12 y 15 años, siendo 34 mujeres y 37 hombres.
- 86 estudiantes de octavo; tercer año de educación básica secundaria entre los 13 y 15 años, siendo 40 mujeres y 46 hombres; y finalmente, 67 de grado noveno; de cuarto año de educación básica secundaria entre los 14 y 17 años, siendo 43 mujeres y 34 hombres.

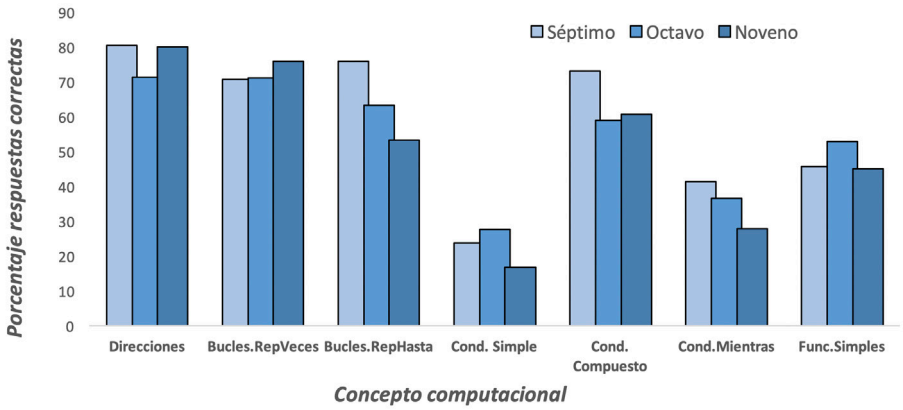
Los resultados finales, fueron analizados mediante estadística descriptiva a través de porcentajes de aciertos, presentándose gráficamente el desempeño en el test, a nivel de las tareas y conceptos computacionales donde alcanzaron mayor desempeño y dificultad.

Resultados y discusión

- **Conceptos computacionales:**

En los resultados obtenidos del test se observa que el 52,51% de los estudiantes logró resolver de forma adecuada la mayoría de los ítems propuestos en el cuestionario. En los conceptos: *direcciones, bucles repetir varias veces, bucles repetir hasta y condicional compuesto* se registraron los mayores porcentajes (52%,51%; 77,5%; 72,7%; 64,3% y 64,4%) de estudiantes que resolvieron de forma adecuada los ítems correspondientes a cada concepto (Figura 1).

Figura 1. Desempeño de los estudiantes en los diferentes conceptos computacionales evaluados.



Fuente. Elaboración propia (2020)

Al analizar lo anterior, es posible afirmar que una de las razones por las cuales *direcciones* es el concepto en el que los estudiantes presentan menor dificultad está relacionada con su carácter elemental en la programación con *Scratch*; es el primer concepto abordado en los ejercicios de introducción a *Scratch*; es un concepto que se aborda en todos los años de instrucción y también en otras asignaturas (matemática, ciencias naturales) dado que permite al estudiante definir la posición de objetos en función de las coordenadas X e Y. Aunque no es citado en el modelo

MIT-Harvard, fue evaluado como punto de partida y concepto fundamental para la programación con *Scratch*.

A diferencia de lo reportado por Grover y Basu (2017), en el análisis general, la mayoría de los estudiantes no presentó dificultades en el concepto *bucles* (*'loops'*). Sin embargo, al revisar el desempeño en cada una de las preguntas que evaluaban dicho concepto, se observó que en el caso de *bucles repetir varias veces,* la mayoría de los participantes (63%) tuvo dificultad en la pregunta 8 que requería un poco más de atención e involucraba una secuencia de código más larga que las demás preguntas. Una situación similar se observa en la pregunta 12, que evalúa el concepto *bucles repetir hasta*, donde solo tres estudiantes del grado octavo y seis del grado noveno lograron resolverla correctamente. Pese a que en el grado séptimo la mayoría (70%) logró resolverla de forma adecuada, el 75% de los estudiantes que desarrollaron el test presentó dificultades en ese ítem.

En ese contexto, atendiendo a las características del cuestionario, en el que los ítems para '*Secuencias y Bucles'* es, en comparación al resto de ítems, de menor carga cognitiva y de mayor carga atencional (Román-González, 2016), las consideraciones de Grover y Basu (2017) son plausibles y los resultados de test no son concluyentes respecto del dominio del concepto. *Bucles* es un concepto en el que los estudiantes deben especificar cuántas veces se ejecutarán las instrucciones o cuándo se detendrá la repetición lo cual implica expresiones aritméticas o condicionales que integran variables. En algunos casos, cuando el valor de la variable cambia a medida que se ejecuta el ciclo, los estudiantes tienden a malinterpretar la situación.

En el caso del concepto *condicionales*, los estudiantes tuvieron mejor desempeño en las preguntas que evaluaban condicional compuesto que en las de condicional simple; *condicional simple* fue el concepto donde los estudiantes tuvieron menor porcentaje (23%) de respuestas correctas (Figura 1), resultado que llama la atención reconociendo que es el concepto de menor complejidad entre los condicionales. Esta es una situación en revisión para la cual aún no se ha definido una explicación sólida. De momento, se están revisando aspectos metodológicos y los resultados a nivel de cada pregunta.

Por ello, a continuación se revisará lo relacionado con los resultados en *condicional mientras que* y *condicional simple,* donde se observó desempeño bajo según la cantidad de estudiantes que resolvieron de forma acertada las preguntas propuestas (35% y 23% respectivamente). También porque, los investigadores, creadores del cuestionario reconocen que los ítems que evalúan estos conceptos, constituyen

una subescala *'Condicionales'* que presenta una fiabilidad suficiente con ítems de dificultad media-alta.

Condicionales es un concepto relacionado con el pensamiento lógico y se centra en la capacidad de tomar decisiones basadas en ciertos estados o situaciones. Se utiliza para indicarle al computador que debe evaluar una condición y, a partir del resultado, ejecutar el bloque de instrucciones correspondiente. La ejecución de condiciones (estructuras de selección) está asociada con la aptitud verbal (Howland y Good, 2015), la percepción y atención, lo que implica una mayor exigencia desde un punto de vista perceptivo-atencional, que la ejecución de secuencias y bucles (estructuras de secuenciación y estructuras de repetición) (Mühling et al., 2015).

Funciones es otro de los conceptos en donde un bajo porcentaje (48%, Figura 1) de estudiantes logró resolver adecuadamente las preguntas. De acuerdo con Román-González, (2016) los ítems de esta sub escala tenían un nivel de dificultad media-baja y en su solución se conjugan la dimensión cognitiva con la exigencia de persistir en el esfuerzo. Así, además de reconocer dificultades propias de los procesos cognitivos asociados a dicho concepto no se descartan cuestiones propias de la concentración e interés.

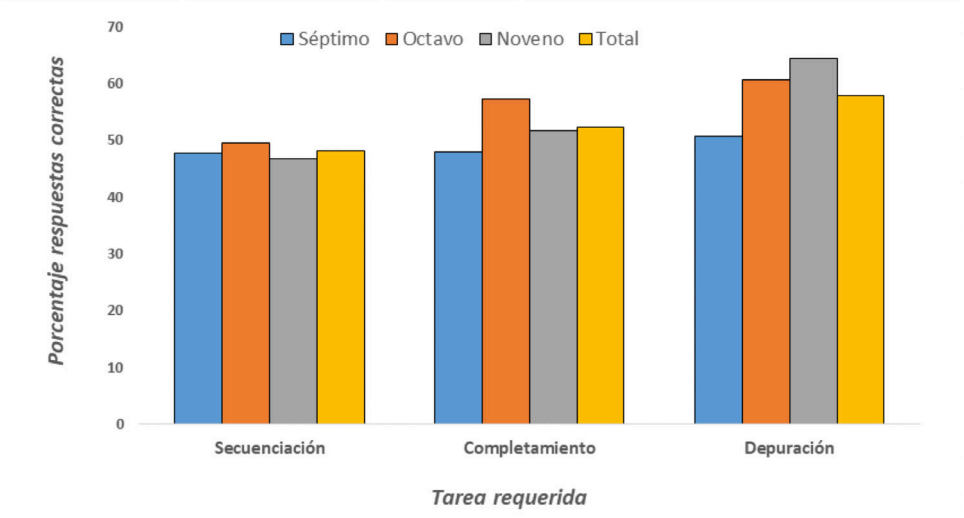
Desde una perspectiva general, a nivel de los grados escolares, el grado séptimo tuvo mejor desempeño en toda la prueba; el grado octavo solo en los conceptos condición simple y función simple. El grado noveno alcanzó mayor porcentaje de respuestas adecuadas en los conceptos *direcciones, bucles repetir hasta y condicional compuesto* (Figura 1). Como se indicó en la metodología, los estudiantes del grado noveno han participado por más tiempo en las actividades de instrucción básica sobre Scratch, por esto, se esperaba que tuvieran mayor número de respuestas correctas que los demás grupos, y que los estudiantes de grado séptimo tuvieran mayor dificultad. Dados los resultados, es necesario revisar con mayor detalle aspectos del desempeño general de cada grupo en otras asignaturas, así como cuestiones relacionadas con los métodos de instrucción y trabajo que se han desarrollado.

- **Tareas computacionales requeridas:**

En relación con las tareas requeridas, el porcentaje general de estudiantes que resolvió de forma adecuada los ítems propuestos fue superior al registrado en los conocimientos. En la tarea *secuenciación* que implica enunciar de manera ordenada una serie de comandos-órdenes, hubo mayor porcentaje (52%) de estudiantes con dificultades y el menor porcentaje se observó en la tarea *depuración* (42%), es decir, *depurar* ('*debug*') un conjunto incorrecto de comandos previamente dado es la tarea

donde el grupo en general alcanzó mejor desempeño. En las tareas *secuenciación* y *completamiento* (completar un conjunto incompleto de comandos previamente dado), los estudiantes del grado octavo alcanzaron mejor desempeño que los del grado séptimo y noveno. Los últimos, se desatacaron en la tarea depuración; el 64,5 %, resolvió adecuadamente los enunciados que la involucraban (Figura 2). A diferencia de los conceptos, en las tareas el grado séptimo tuvo mayores dificultades. Así, en este componente del pensamiento computacional (tareas), se observan resultados menos favorables que los observados en los conceptos computacionales. Aunque se tiene un perfil de las habilidades de los estudiantes a nivel de conceptos y tareas cognitivas requeridas es necesario hacer un análisis específico a nivel de cada pregunta de manera que se puedan identificar posibles tendencias o relaciones entre las respuestas para los conceptos y tareas, también analizar casos particulares de estudiantes aventajados o con dificultades.

Figura 2. Desempeño de los estudiantes en las tareas computacionales evaluados



Fuente. Elaboración propia (2020)

De acuerdo con la literatura, los resultados no distan de lo reportado en otros estudios a nivel de dificultades en habilidades computacionales asociados a la programación con *Scratch*. Pueden estar asociados con las características mismas del uso de programación a través de bloques, la limitada formación docente y experiencia que se tiene en la enseñanza del pensamiento computacional así como otros factores que seguirán siendo explorados.

En tal sentido es básico para fortalecer el proyecto curricular y de investigación que se adelanta tener en cuenta aspectos propios de la programación con bloques, hasta ahora no considerados, que han surgido en el proceso de análisis de la información recolectada y que son coherentes con lo propuesto por Lee (2010):

> a. El uso de bloques *para siempre*, para un controlador de eventos es una característica única de *Scratch* y los estudiantes a menudo olvidan incluir el bucle infinito para activar los códigos de manejo de eventos, por ello se debe prestar especial atención a las explicaciones sobre la función de los bucles en ese contexto particular. Es importante también, proporcionar una demostración clara de cómo los controladores de eventos integrados son equivalentes a otras formas de hacer la función, para que los estudiantes puedan evaluar varios métodos de computación.
> b. Para utilizar *Scratch* en la transición a los lenguajes de programas comunes basados en texto, se debe enfatizar las características únicas de *Scratch*. *Repetir hasta*, por ejemplo, también deben compararse con otra sintaxis de programación, como un bucle *mientras* (*while).* En *Scratch*, el bloque de *repetición hasta* ejecuta su código hasta que la condición se vuelva verdadera. En contraste, los bucles *while* (*mientras*) ejecutan su código cuando la condición especificada es "verdadera" y detienen su ejecución una vez que se vuelve "falsa".
> c. Enfatizar la capacidad de descomponer problemas y diseñar soluciones. Las formas en que se utilizaron los bloques definidos por el usuario revelaron la competencia de los estudiantes para la descomposición del problema.

Conviene no solo analizar con mayor detalle los resultados del test y evaluar los prototipos de videojuegos creados por los estudiantes mediante el uso de *Dr. Scratch* y matrices específicas, sino también promover la discusión y socialización de las creaciones de los estudiantes. Al igual que Arzarello et al. (1993), a nivel de la propuesta curricular y de investigación, se reconoce que la comprensión de los conceptos es necesaria pero no suficiente para desarrollar un programa efectivo; el análisis de los programas de los estudiantes es crucial para evaluar la internalización de los conceptos computacionales.

Es importante en esta dirección, reconocer que el entrenamiento en programación es básicamente entrenamiento en pensamiento y no debe esperarse que el uso de la computadora desarrolle estas habilidades (Oluk y Korkmaz, 2016). De esta forma, resulta fundamental revisar resultados de programación con otras aplicaciones, fortalecer otros ejercicios no centrados en el uso del computador como los ejercicios que se proponen desde preescolar en programas internacionales de pensamiento

computacional: laberintos, bloques lógicos, juegos no virtuales, el fortalecimiento de ejercicios de organización de secuencias (diagramas de flujo, de relaciones, etc.).

Con lo anterior se pretende atender ciertas limitaciones planteadas sobre la programación con bloques, específicamente con *Scratch,* permitiendo a los estudiantes definir sus propios bloques, lo que les faculta para personalizar códigos complejos en un bloque reutilizable aunque los estudiantes a menudo no usan los bloques definidos por el usuario (Moreno y Robles, 2014).

Por su parte, Meerbaum-Salant et al., (2013) consideran que *Scratch* puede mejorar la comprensión de conceptos de programación; es un proceso de desarrollo totalmente ascendente que comienza con los bloques individuales y tiene una tendencia a la programación extremadamente fina.

Consideraciones finales

Aunque no son concluyentes, los resultados del test indican mayor dificultad de los estudiantes a nivel de conceptos computacionales que en las tareas, principalmente en los conceptos *condicional simple* y *condicional mientras que*. A nivel de las tareas, s*ecuenciación,* es donde se observaron más dificultades.

No es evidente una diferencia en las habilidades computacionales evaluadas (conceptos y tareas) entre los diferentes grados. Frente a ello, podría suponerse que la instrucción presentada a través de los años debe replantarse y fortalecerse mediante acciones como el fortalecimiento de la formación docente, el ajuste de las acciones didácticas que se llevan a cabo en el aula así como los procesos curriculares institucionales.

Los resultados del estudio así como el proceso, ampliaron las perspectivas hacia futuras investigaciones; permitieron identificar otros elementos de análisis para los resultados del test, por ejemplo, análisis específico para grupos de estudiantes con mayor dificultad, relaciones entre el desempeño a nivel de conceptos y tareas, así como la relación entre los resultados del test y las características de los prototipos de videojuegos propuestos por los estudiantes. Ponen de manifiesto la necesidad de explorar otros análisis que permitan identificar con mayor especificidad conceptos, subprocesos u otros elementos propios del pensamiento computacional que se deben fortalecer.

La reflexión a nivel de la experiencia curricular, de trabajo de aula junto con lo ya explicitado desde el componente investigativo, evidencian la necesidad de fortalecer el trabajo interdisciplinar que podría tener como punto de partida el modelo transversal de desarrollo del pensamiento computacional propuesto por Barr y Stephenson (2011). Esto quiere decir que es necesario revisar con mayor detalle la planeación curricular para que se aborden desde las diferentes asignaturas otros componentes del pensamiento computacional no considerados en el presente análisis y con ello, se fortalezca el pensamiento lógico, algorítmico, causal y creativo; la abstracción, la organización de datos, etc.

Desde una perspectiva general, pese a la inexperiencia e inequidad tecnológica, denominada así por las limitaciones en cuanto a formación de los docentes, la disposición de espacios y recursos para la capacitación, la infraestructura, elementos de apoyo como bloques lógicos para trabajar en preescolar y conectividad, se han dado grandes pasos y se cuenta con resultados para repensar y reestructurar las acciones de aula.

La propuesta que se adelanta constituye un desafío en un contexto de inequidad tecnológica en Colombia, dado que se atiende un reto que viene de varias décadas atrás y que se hace más retadora en la medida que avanza a pasos agigantados y los espacios de tiempo y capacitación con los que se dispone son limitados.

Referencias bibliográficas

Arzarello, F., Chiappini, G. P., Lemut, E., Malara, N., y Pellerey, M. (1993). Learning Programming as a Cognitive Apprenticeship Through Conflicts. En *Cognitive Models and Intelligent Environments for Learning Programming* (pp. 284-298). Berlin, Heidelberg: Springer.

Barr, V. y Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community?. *ACM Inroads*, 2 (1), 48-54.

Brennan, K., y Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Comunicación presentada en Annual Meeting of the American Educational Research Association, Vancouver: Canada. Recuperado de http://Scratched.gse.harvard.edu/ct/files/AERA2012.pdf.

Buss, A., y Gamboa, R. (2017). Teacher Transformations in Developing Computational Thinking: Gaming and Robotics Use in After-School Settings. En *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 189-203). Cham: Springer International Publishing.

CAS, Computing at School. (2015). *Barefootcomputing*. https://www.barefootcomputing.org/curriculum

Daza-Pérez, E. y Santoyo, F. (2016). Construir videojuegos con Scratch para fortalecer habilidades de pensamiento creativo. Aproximaciones a partir de una experiencia en el contexto rural. En: Legerén-Lago, B y Crespo-Pereira, V. "*De la Idea a la Pantalla. Compendio de Investigaciones sobre juegos serios*" (pp. 33-43).Vigo: Universidad de Vigo:.

González, R. (2016). *Código alfabetización y pensamiento computacional en educación primaria y secundaria: validación de un instrumento y evaluación de programas* (Tesis doctoral, Universidad Nacional de Educación a Distancia, España). Recuperado de http://e-spacio.uned.es/fez/view/tesisuned:Educacion-Mroman.

Grover, S. y Basu, S. (2017). Measuring Student Learning in Introductory Block-Based Programming: Examining Misconceptions of Loops, Variables, and Boolean Logic. En *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, (pp. 267–272). Recuperado de https://dl.acm.org/doi/10.1145/3017680.3017723.

Howland, K., y Good, J. (2015). Learning to communicate computationally with flip: A bi-modal programming language for game creation. *Computers & Education,* 80 (January), 224-240.

Lee, Y. (2010). Developing computer programming concepts and skills via technology-enriched language-art projects: A case study. *Journal of Educational Multimedia and Hypermedia*, 19 (3), 307-326.

Lee, I. y Malyn-Smith, J. (2019). Computational Thinking Integration Patterns Along the Framework Defining Computational Thinking from a Disciplinary Perspective. *Journal of Science Education and Technology*, 29, 9–18.

López, J. (Agosto de 2011). Programación con Scratch [Documento pdf en página web]. Recuperado de http://eduteka.icesi.edu.co/pdfdir/AlgoritmosProgramacionCuaderno1.pdf

Maloney, J., Resnick, M., Rusk, N., Silverman, B., y Eastmond, E. (2010). The Scratch programming language and environment. ACM Trans. *Comput. Educ.* 10, (4), 1-15.

Meerbaum-Salant, O., Armoni, M., y Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education*, 23 (3), 239-264.

Ministerio de Educación Nacional (2018). *Ser competente en tecnología: ¡Una necesidad para el desarrollo!, Serie guías N.30.* Bogotá: Ministerio de Educación Nacional.

Moreno, J., y Robles, G. (2014). Automatic detection of bad programming habits in Scratch: A preliminary study. En *Proceedings IEEE Frontiers in Education Conference, FIE* (pp. 1-4). Madrid: IEEE Computer Society.

Oluk, A. y Korkmaz, Ö. (2016). Comparing Students' Scratch Skills with Their Computational Thinking Skills in Terms of Different Variables. *I.J. Modern Education and Computer Science*, 11, 1-7.

Reding, T. E., Dorn, B., Grandgenett, N., Siy, H., Youn, J., Zhu, Q., y Engelmann, C. (2016). *Identification of the Emergent Leaders within a CSE Professional Development Program*. Comunicación presentada en el 11th Workshop in Primary and Secondary Computing Education, Münster: Germany. Recuperado de https://www.researchgate.net/publication/317012225_Identification_of_the_Emergent_Leaders_within_a_CSE_Professional_Development_Program.

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

Sáez-López, J.-M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two-year case study using "Scratch" in five schools. *Computers y Education*, 97, 129-141.

Vázquez, E. A., Bottamedi, J., y Brizuela, M. L. (2019). Pensamiento computacional en el aula: el desafío en los sistemas educativos de Latinoamérica. *Revista Interuniversitaria de Investigación en Tecnología Educativa*, 7, 36-47.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM,* 49 (3), 33-35.

Zapata-Ros, M. (2015). Pensamiento computacional y alfabetización digital. *RED, Revista de Educación a Distancia,* 46 (4), 47 pp. Recuperado de http://www.um.es/ead/red/46/zapata.pdf.