Block-Based Design: Un Método Para El Diseño de Programas Basados en Bloques

Javier Ricardo Muñoz Castillo

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Sistemas

Línea de Investigación en Ingeniería del Software

Popayán, noviembre 2022

Block-Based Design: Un Método Para El Diseño de Programas Basados en Bloques

Monografía como requisito para optar al título de Ingeniero en Sistemas

Director

Ph.D. Julio Ariel Hurtado Alegría

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Sistemas

Línea de Investigación en Ingeniería del Software

Popayán, noviembre 2022

Contenido

Lista de figuras

Lista de tablas

Introducción

En este documento se exponen los contenidos teóricos y conceptuales que permiten consolidar una propuesta basada en la aplicación de un método que facilite el pensamiento computacional a través del despliegue de mecanismos de abstracción, que se concreten en un diseño de programas basados en bloques, que al ser descompuestos e integrados en nuevas estructuras permitan a los niños la construcción de programas específicos.

Por tanto, en primer término, se abordan el planteamiento del problema, los objetivos y la justificación; luego, se establece el marco referencial con sus componentes: marco teórico y marco contextual.

Posteriormente, se establecen los aspectos metodológicos y los momentos de investigación: exploración, formulación, evaluación y documentación; igualmente se establece la manera en que se recolectar la información mediante una estrategia de búsqueda estructurada en fuentes bibliográficas, con base en palabras clave predefinidas, que sirven de sustento al desarrollo de la monografía.

Después, se despliegan cada uno de los capítulos referentes a los objetivos específicos planteados: Se caracterizan las estrategias de diseño software en el contexto del desarrollo del pensamiento computacional en niños por medio de una revisión sistemática de la literatura.

Entonces, se propone el empleo de un método de diseño software que facilite la manipulación abstracta de los bloques mediante su construcción, descomposición e integración en nuevos bloques, con propósitos específicos, y siguiendo la caracterización previamente realizada; como tarea final, se evalúan las actividades realizadas por los estudiantes del caso exploratorio en concordancia con la metodología elegida y un lenguaje de programación basado en bloques.

Finalmente, se realizan las conclusiones, recomendaciones y se despliega la bibliografía.

1. Planteamiento del problema

Cuando se interactúa con niños es necesario tener presente que los adultos se enfrentan a un mundo totalmente diferente, en el que se encuentra una variedad de estructuras mentales y representaciones de la realidad de acuerdo a las experiencias y percepción de los sentidos en el niño (Bautista et al., 2009).

Más aún, cuando se trabaja en informática, se enfrentan inconvenientes de comprensión de problemas, obstáculos que brindan un espacio educativo rico en oportunidades para entender las estructuras mentales de los niños y aplicar los principios de ingeniería.

En la actualidad muchas iniciativas de trabajo de la programación con niños usan el paradigma de programación basada en bloques debido a que "existe un reconocimiento creciente que la computación es una competencia esencial para que todos los estudiantes la desarrollen para tomar parte activa en un mundo cada vez más digital" (Weintrop & Wilensky, 2017, p.1).

Este tipo de programación, a diferencia de la basada en líneas de código, permiten la programación visual, mediante la utilización de nuevas herramientas, como Scratch, facilitando en las aulas de secundaria y en las áreas de introducción a la informática, el desarrollo del pensamiento computacional (Weintrop & Wilensky, 2017).

Sin embargo, de acuerdo a experiencias previas en Child Programming (Belalcázar, 2018; Hurtado et al., 2011; Rich et al., 2019), los niños se enfrentan a inconvenientes relacionados con la forma en que deben usar la abstracción y la descomposición desde un enfoque top-down, dificultando, por tanto, el análisis de problemas y el diseño de las soluciones.

Los trabajos previos realizados por el grupo IDIS (Zúñiga et al., 2016) han evidenciado el valor de las abstracciones, pero aún se carece de la forma en que los niños puedan ser guiados para obtenerlas, por lo que descomponer un problema o solución sigue siendo la parte más difícil para aprovechar el trabajo colaborativo.

Por otra parte, el trabajo de (Dietz et al., 2019), plantea una estrategia de representación visual para mejorar el pensamiento computacional y su relación en el éxito de solución de problemas computacionales con niños, pero los autores indican que descomponer es uno de los problemas más relevantes, y que a la fecha no se cuenta con una guía que les indique a los niños cómo usar la descomposición para plantear una solución.

(Rich et al., 2019) avanza en la comprensión de la descomposición como habilidad del pensamiento computacional y la describen como un proceso de categorizar elementos potenciales como identificar elementos sustantivos, identificar relación entre elementos, emplear diferentes estrategias para realizar la descomposición y evaluar iterativamente la utilidad de descomposición, indicando que se requiere mayor exploración en la investigación sobre este problema. Sin embargo, los autores dejan planteada una pregunta abierta: ¿cómo se puede operacionalizar la descomposición de la forma más adecuada?

Por lo anterior, en este proyecto de grado se plantea la siguiente pregunta de investigación: ¿Cómo guiar a los niños en el diseño de software considerando abstracciones de alto nivel que faciliten la descomposición de soluciones a ser implementadas bajo el paradigma de programación basado en bloques?

2. Justificación

Tradicionalmente los principios y prácticas del desarrollo de software pertenecían a los ingenieros de software en la industria, no obstante, con el avance y la acogida de la tecnología en la vida diaria, su desarrollo ha terminado involucrando en forma directa a las personas del común como niños y adolescentes.

Así, en el ámbito educativo la robótica en el aula es parte de la alfabetización computacional desde los estados iniciales de la educación, a edades tan tempranas como los tres años, haciendo patente la necesidad de analizar las dificultades y los beneficios de llevar la Internet de las cosas a estos niveles, pero con la convicción que desde aquí inicia la construcción de las competencias en abstracción y de la utilización del pensamiento abstracto, clave para el pensamiento computacional (Kramer, 2007);(Berciano-Alcaraz et al., 2022).

Así, es importante enfatizar en el desarrollo del pensamiento computacional tanto como leer, escribir y las matemáticas, permitiendo desarrollar la capacidad de análisis y representación de problemas reales y plantear soluciones computacionales; de hecho, el pensamiento computacional "se construye sobre los límites y la potencia de los procesos computacionales, realizados por autómatas o por humanos. Los métodos y modelos computacionales nos proporcionan la motivación para resolver problemas y diseñar sistemas" (Wing, 2006) .

Por tanto, el pensamiento computacional deviene una importante herramienta cognitiva que se puede desarrollar en todas las áreas de la educación, si bien su construcción tiene aspectos aún desconocidos lo que dificulta su medición, particularmente cuando se trata de la acción de descomposición, que permite dividir un problema en sus componentes, de manera que sea efectiva la resolución de problemas complejos (Rich et al., 2019).

La programación basada en bloques en la actualidad es uno de los enfoques metodológico más populares para la enseñanza de la programación inicial, utilizándose en interacción con contextos de micromundos, resolviendo acertijos específicos, a fin de mejorar competencia de razonamiento lógico, de pensamiento computacional, de una forma mucho más atractiva y motivante (Pelánek & Effenberger, 2022).

Es por eso que este trabajo de grado busca proponer y evaluar un método de diseño de software que le permita a un equipo de niños manejar la complejidad para resolver problemas, mediante el uso del pensamiento computacional e incentivar al desarrollo de software, aportando un método guía para la educación en Colombia y promover un aprendizaje significativo en las escuelas.

3. Objetivos

3.1. Objetivo general

Proponer y evaluar un método para el diseño de programas basados en bloques, utilizando mecanismos de abstracción, propios del pensamiento computacional, para facilitar a los niños la construcción, descomposición e integración de nuevos bloques.

3.2. Objetivos específicos

1. Caracterizar las estrategias de diseño de software en el contexto del desarrollo del pensamiento computacional en niños mediante la revisión de la literatura.
2. Proponer un método de diseño de software que permita integrar mecanismos de abstracción para facilitar a los niños la construcción, descomposición e integración de nuevos bloques considerando las estrategias caracterizadas.
3. Evaluar el método de diseño en términos de correspondencia entre los modelos de diseño elaborados y los programas elaborados en el lenguaje basado en bloques.

4. Marco Referencial

En esta capitulo se expone los conceptos teóricos fundamentales relacionados con el pensamiento computacional, la capacidad de abstracción, la resolución de problemas, los modelos mentales y la manera en que se puede relacionar el diseño de software con las herramientas de programación visual aportadas por el diseño basado en bloques (Block-based Design), particularmente en los entornos educativos iniciales en el área de informática.

De igual forma, se discute un probable contexto educativo de aplicación donde se piensan implementar las pautas contempladas dentro de la propuesta de esta monografía de investigación.

4.1. Marco teórico

4.1.1. Pensamiento Computacional

El pensamiento computacional es tan importante que se han creado retos como el concurso Bebras con el objetivo primordial de motivar a los estudiantes a que se interesen por las temáticas de informática y promover el pensamiento de tipo algorítmico, lógico, operacional y basado en la informática básica; la idea es motivar a los niños para que aprendan los conceptos con temática y apoyarlos en el desarrollo del pensamiento computacional a través de pruebas y quizzes que les permiten evaluar si en corto tiempo los estudiantes puede llegar a soluciones correctas (Laanpere & Kori, 2020).

De igual manera, en los ámbitos escolares se gestan cambios en el currículo escolar promoviendo la idea de la alfabetización digital en Informática y TIC (tecnologías de la información y la comunicación), para lo cual se generaliza cada vez más la implementación del pensamiento computacional en los currículos de preprimaria, primaria y los cursos iniciales y superiores de la secundaria, implicando cambios en el entrenamiento de los profesores, desarrollo de materiales de aprendizaje con nuevos temas, enfocados en robótica, conceptos teóricos de informática y programación por bloques con Scratch (Laanpere & Kori, 2020).

Asimismo, un mundo digital exige el énfasis en las disciplinas denominadas STEM (ciencias, tecnología, ingeniería y matemáticas) y por ello dentro de los fundamentos para el desarrollo del pensamiento computacional pueden citarse aquellas que utilizan en primer término los objetos y los micromundos, para luego introducir el pensamiento computacional y la programación,

privilegiando experiencias fenomenológicas que potencian las experiencias positivas, que privilegian en la abstracción, lo que exige que por pensamiento computacional se entienda como:

> Todo conjunto de ideas computacionales que adquieren las personas a través de su trabajo para diseñar programas, software, que va a ser ejecutado en el hardware de una computadora; es el conjunto de procesos de pensamiento que se involucran en la formulación del problema de tal forma que se pueden representar sus soluciones mediante pasos y algoritmos que se van a realizar de forma efectiva por un agente de procesamiento de información. (Khine, 2018)

Entonces, como el pensamiento computacional está considerado ya un elemento central en todas las disciplinas STEM, se puede definir como el proceso de solución de problemas de manera que se pueda permitir usar equipos informáticos, organizar datos de manera lógica, representar las soluciones mediante abstracciones y transferir este proceso en cualquier tipo de problema cotidiano (Carnevale et al., 2011).

El pensamiento computacional, sea considerado como conjunto de procesos de pensamiento que sirven a la formulación y expresión de un problema en un formato que pueda ser procesado por una máquina, sea considerado como actividad mental para abstraer problemas y generar soluciones que se puede automatizar, está compuesto por conceptos claves como pensamiento lógico, pensamiento algorítmico, descomposición, generalización y reconocimiento de patrones, modelización, abstracción y evaluación (Beecher, 2017).

En este orden de ideas, el pensamiento computacional ha despertado el interés y ha generado un gran esfuerzo para incorporarlo a través de proyectos, juegos, entornos de programación en el ámbito de las escuelas y universidades, dirigiéndose los principales esfuerzos a consolidarlo en niños, adolescentes y jóvenes, utilizando herramientas de programación visual basadas en bloques como Scratch.

Scratch es un lenguaje de programación creado por el MIT y especialmente diseñado para que todos puedan iniciarse en el mundo de la programación, abordar situaciones hipotéticas, dar soluciones a problemas cotidianos, pero no constituyen la verdadera realidad (Senge, 1990). Mediante esta plataforma de programación abierta, los estudiantes son capaces de adquirir o mejorar habilidades y capacidades para comprender, diseñar y construir proyectos, utilizando técnicas y estructuras que

se usan en la ciencia de la computación (Basogain et al., 2015).

4.1.2. Modelos Mentales

Cuando se habla de modelos mentales se deben tener en cuenta tres dimensiones fundamentales que los caracterizan: la naturaleza del dominio de conocimiento estudiado; el tipo de aproximación teórica; la naturaleza de la metodología empleada. Los dominios de conocimiento son o fácticos o de tipo matemático; el enfoque teórico refiere a las representaciones del conocimiento enfocadas desde la semántica computacional como ha sido desarrollada por la inteligencia artificial; las metodologías son de tipo ecléctico que incluyen comparaciones del modelo de simulación con la realidad, estudios de desempeño, desarrollo, entre otras (Gentner & Stevens, 1983).

Por tanto, los modelos mentales implican el estudio del razonamiento y de las habilidades que las personas deben poseer para desempeñar tareas cognitivas, utilizando la abstracción para enfrentar situaciones problemáticas planteadas por diversas tareas del entorno, enfatizando que los pensamientos son unidades fundamentales de la experiencia puesto que los fenómenos mentales son intencionales y orientados al objeto, marcándose así límites específicos entre el pensamiento y la percepción, implicando que cualquier aspecto de la cognición se relaciona con los procesos de inferencia (Manktelow & Chung, 2004).

De hecho, al hablar de inferencia se hace referencia a la deducción como modo en que el razonamiento examina el estado de las cosas que se ha afirmado en las premisas para formar un diagrama en el que se establecen relaciones, que conduzcan a una función necesaria o probable una verdad, lo que evidencia que desde la percepción se produce un modelo mental, que también surge de la comprensión lingüística, estableciendo que el pensamiento y razonamiento son manipulaciones internas de modelos mentales(Manktelow & Chung, 2004).

En este sentido, la naturaleza de la explicación se relaciona con su poder para predecir eventos ya que puede traducir un proceso externo en palabras, números u otros símbolos que pueden ser utilizados como un modelo del mundo y de hecho, llevar una pauta de razonamiento desde estos símbolos hacia otros, resignificándolos, de manera que se va de los símbolos hacia los procesos externos o al menos al reconocimiento que la simbología corresponde a este tipo de procesos (Johnson-Laird, 2004).

Entonces, los modelos mentales son de orden simbólico, icónico; implican una teoría en la que el

modelo refiere un sistema que tiene una estructura relacional semejante al proceso que imita, hacen semejanzas con la realidad, y por tanto representan razonamientos proposicionales, estrategias, contraejemplos (Johnson-Laird, 2004).

De allí que, los modelos mentales sean representaciones prototípicas que explotan la percepción visual de la realidad, de tal forma que debe cumplir las reglas de inferencia lógica para tomarse como verdaderos y ser coherente con lo que se conoce acerca de las cosas que modela en el entorno externo; plantean entonces tres cuestiones fundamentales: definir si las imágenes que los representan difieren de las proposiciones; establecer las reglas de inferencia en la que se sustentan sus razonamientos y los procesos mentales involucrados; comprender la manera en que se establecen los significados de sus palabras y si estos están en función de una descomposición o de postulados (Johnson-Laird, 1980).

Por otra parte, partiendo del hecho que la cognición depende de las experiencias recibidas a través del cuerpo, mediante la percepción, medidas por contextos biológicos, culturales, sensorio motrices y psicológicos, que devienen en acción, se concluye que la percepción y la acción contribuyen a la consolidación de estructuras cognitivas que se traducen en conductas, en modelos mentales que incorporan la acción derivada de las experiencias obtenidas a través de la percepción (Varela et al., 1991).

De allí que, los modelos mentales sean supuestos e historias, narrativas, representaciones del mundo, que yacen en la mente y que están en función de la percepción del mundo externo, basadas en las interacciones con este, con la experiencia sensible basada en lo corpóreo, de manera que son dinámicos, cambiantes y determinan la manera de actuar de cada persona (Senge, 1990). Además, los modelos mentales, como representaciones simbólicas de la realidad, que se produce a nivel cognitivo, interno, poseen un significado en función de las experiencias y estilos de vida.

Sintetizando las ideas anteriores, se puede decir que los modelos mentales son representaciones personales, simbólicas, icónicas, prototípicas, dinámicas, internas, basadas en la experiencia adquirida a través del cuerpo, mediante las percepciones, estructuradas con la ayuda de imágenes, razonamientos, deducciones, proposiciones, contextos psicológicos, biológicos y culturales, que sirven paras la interacción con el mundo exterior; “son utilizados para razonar y tomar decisiones, y pueden ser la base de las conductas individuales. Proporcionan los mecanismos mediante los cuales la nueva información es filtrada y almacenada”(Jones et al., 2011).

4.1.3. Abstracción

Al considerar la inteligencia como el control de la conducta mediante sistemas cognitivos, al estado de desarrollo que se caracteriza por la incorporación de estructuras que permiten un mayor control de la conducta mediante las formas lógico matemáticas propias del ser humano, se le denomina abstracción reflexiva, que permite la transición del dominio de los sentidos al del pensamiento, creando relaciones entre los contenidos cognitivos y las representaciones lógico-matemáticas, de modo que la abstracción es la construcción que permite estas relaciones entre lo sensible y los modelos mentales (Damerow, 1996).

Luego, el término abstracción implica, en sentido general, simplificación o eliminación de algún detalle sin importancia, pero, en el dominio de la ciencia de la computación, una abstracción que es un "concepto intelectual para simplificar y eliminar factores que no son importantes con respecto a la idea clave […] con la finalidad de hacer más simple los procesos de pensamiento y el desarrollo de sistemas" (Keller, 2001, p.1). Esto conlleva mirar los sistemas como un conjunto de componentes que interactúan de la forma más simple posible, es decir, con diferentes niveles de abstracción para facilitar su implementación.

Entonces, la abstracción computacional se inspira en la manera en que funciona el cerebro humano, considerado como una computadora muy sofisticada, donde existen diferentes jerarquías de abstracción en función de los distintos niveles de complejidad de la información que se requiere procesar, para lo cual son básicos tres aspectos: definir formalmente el problema a resolver; establecer un algoritmo que lo resuelva; implementar el algoritmo dentro de la estructura computacional que lo va a ejecutar; estos niveles refieren a la creación de programas que controlen cuando y de qué manera se van a hacer los cálculos, que acción se va a llevar a cabo, los datos requeridos, etc. (Ballard, 2015).

Desde la perspectiva etimológica, remite al latín abs-trahere, que deriva del griego aphairesis, y que tiene la connotación de alejarse, apartarse; sin embargo, el término se ha revestido de diversas connotaciones, con el paso del tiempo; según el diccionario de Oxford, abstracción implica: tratar con ideas y no con eventos; representación libre de las cualidades en el arte; el proceso de considerar algo de forma independiente de sus asociaciones o atributos; un estado de preocupación;

el proceso de remoción de algo. Esto demuestra que este vocablo se incorpora al vocabulario de las actividades pervasivas de la actividad humana donde la percepción, la conceptualización y el razonamiento son importantes y por tanto, no existe una definición que englobe todas sus connotaciones (Saitta & Zucker, 2013).

Luego, la computación, como piedra angular de la era de la información y de las aplicaciones software en matemáticas e ingeniería, utilizan de manera indispensable la abstracción para construir modelos, diseños e implementaciones apropiadas, utilizándola como manera de pensar o proceso, como objeto, para concentrarse en los detalles importantes, ir más allá de las reificaciones, destacar relaciones entre cosas y comunidades, porque lo que es abstracto para mi puede ser concreto para otra; de hecho, la abstracción implica quitar detalles, de simplificar, de enfocarse en lo importante; pero, también presupone generalizar para identificar componente claves o comunes, partiendo de instancias o ejemplos específicos (Frorer et al., 1997); (Serna, 2011).

En cuanto al campo computacional el idealismo implícito en la abstracción refiere a que los principios implicados en las leyes abstractas son más importantes que los fenómenos que se perciben mediante la percepción de los sentidos cuando se trata de considerar la realidad, de establecer una equivalencia entre lo bello y lo verdadero, a través de la deducción lógica, porque las estructuras utilizadas en el software son formas ideales, inmutables, independientes de la contingencia deben ser validadas lógicamente (Fazi, 2018).

Así, la abstracción, como forma de pensar, de estructurar, de modelar la realidad, es una habilidad necesaria que debe mejorarse a través de la capacidad para la resolución de problemas y ser una meta de los programas formativos (Boroditsky et al., 2002).

4.1.4. Programación basada en bloques

En la actualidad, las competencias en pensamiento computacional deben promoverse desde los primeros niveles de la educación, ya que la sociedad de la información es mucho más exigente con respecto a las destrezas en informática, particularmente en las que se relacionan con la codificación, tarea en la que se requiere de innovación por parte de los docentes, promoviendo un enfoque ameno de la programación, utilizando didácticas en las que el proceso de aprendizaje se centre en el estudiante para que este no tenga temor de programar (Williams, 2017).

Así, para promover el pensamiento computacional (codificación) y promover el razonamiento

lógico a través de resolución de problemas utilizando actividades alineadas con este pensamiento tales como simulación, abstracción, descomposición del problema, algoritmos y procedimientos, recolección de datos, automatización, representación de la información, paralización y automatización de tareas, es necesario seleccionar una serie de herramientas de programación basadas en bloques.

Las herramientas de programación basadas en bloques se utilizan frecuentemente en los niveles iniciales de la introducción a la ciencia de la computación a todos los niveles de la enseñanza básica y secundaria puesto que son fáciles de usar, cada bloque está escrito en lenguaje natural, se tiene la posibilidad de interacciones de computación mediante arrastre, y son mucho más poderosas que el enfoque convencional basado en texto, debido a que explotan la metáfora de las primitivas como piezas de un rompecabezas, que permite explotar las ventajas de una programación visual (Weintrop & Wilensky, 2015).

De hecho, la programación visual o gráfica se ha desarrollado con el objetivo de minimizar las dificultades en programación de los principiantes, a través de la promoción de la creación de juegos y simulaciones, automatizando determinados tipos de procesos, como el ensamblaje de funciones utilizando solamente un ratón para seleccionar un conjunto de instrucciones y recibir una respuesta de video o audio que le indique si la construcción dada es válida, porque cada bloque posee pistas visuales que indican al usuario donde y como se puede utilizar, como son su forma, color o categoría a la que pertenece, además de proporcionarse en la capacidad de una ejecución visual de los bloques construidos para cumplir una tarea específica (Ricarose, 2007);(Weintrop & Wilensky, 2015).

Entonces, en la programación gráfica basada en bloques, los usuarios pueden manipular y conectar objetos como siempre empiezo rompecabezas para construir sus programas, indicando la forma de estos bloques la sintaxis del lenguaje porque solamente bloques de formas complementarias pueden conectarse de manera que el aprendiz no está forzado aprender sintaxis confusa ayudando a reducir la curva de aprendizaje a la vez que se minimizan los errores; sigue la filosofía de los denominados editores estructurados que utilizan como unidad de composición un nodo en el árbol de sintaxis abstracta; por ello, la programación se convierte en arrastrar bloques hacia una zona de comandos y en agruparlos para formar secuencias, reteniendo la construcción instrucción a instrucción (Weintrop & Wilensky, 2017);(Ricarose, 2007).

En este orden de ideas, la programación basada en bloques es la manera más común mediante la cual los estudiantes entran en contacto con la ciencia de la computación y son introducidos dentro de la programación, si bien existen diferentes perspectivas con respecto a la manera en que se debe presentar los tópicos fundamentales: la forma en que se presentan los conceptos de programación básico después de que hayan sido entendidos los conceptos por los estudiantes; cómo deben ser las prácticas de programación; como la percepción de la programación y la ciencia de computación que tienen los estudiantes (Wyeth & Purchase, 2000).

Sin embargo, existe la necesidad de realizar evaluaciones sobre la asimilación de los conceptos básicos por parte de los estudiantes ya que el código puede funcionar debido a que se han mezclado de forma tosa códigos utilizables a través del retoque, pero esto no garantiza que tenga una comprensión cabal de cómo funciona el código; la depuración lógica o proceso sistemático de identificar por qué existen fallas en el código de cómo subsanarlas, es parte esencial de la resolución de problemas y por tanto de la programación de la ciencia de la computación (Kim et al., 2018).

4.1.5. Método e Ingeniería de Método

La ingeniería de método promueve la idea de crear metodologías para el desarrollo de los sistemas de información, seleccionando y ensamblando partes de métodos, a partir de una base de datos, donde se recopilan las mejores estrategias metodológicas que pertenezcan a un metamodelo dado, con la finalidad de construir un método situacional que permita al ingeniero cumplir con las metas organizacionales o de un proyecto particular; en este contexto son prácticos los estudios de casos para analizar experiencias de metodologías flexibles y ágiles, que pueden madurar y mejorar con el tiempo, en función de las necesidades y las competencias del talento humano (B. Henderson-Sellers, 2006).

Entonces, la investigación y el desarrollo en el campo de la ingeniería de método se interesan por diseño, construcción y evaluación de métodos, técnicas y herramientas de soporte para el desarrollo de los sistemas de información, explotando las mejores opciones de diversos paradigmas mediante diversos marcos metodológicos de desarrollo que se actualizan constantemente para estar al tanto de los cambios que suceden en la industria, con la ayuda de los avances tecnológicos, aplicando nuevos métodos que tienden a ser modulares con énfasis en componentes genéricos intercambiables, privilegiando marcos de referencia metodológicos como la flexibilidad, la flexibilidad controlada y el control, así como los enfoques cliente -servidor (Brinkkemper et al.,

1996)

Por método de desarrollo de software se comprende una metodología en la que se tiene un proceso disciplinado de desarrollo software para hacer esta tarea más predecible y eficiente, cumpliendo con los requerimientos del cliente, énfasis que ha llevado a cambios profundos en los paradigmas, que van desde la utilización del lenguaje unificado de modelado (UML), hasta las metodologías Lean o ágiles, que han tomado relevancia en las últimas décadas, tanto que, son aceptadas en otras disciplinas como la gestión de proyectos, por su habilidad para desarrollar el software de forma eficiente, flexible y en poco tiempo(Gacitúa-Bustos, 2014).

Así las cosas, en este proyecto, se utilizan las ideas de las metodologías ágiles para articular los elementos de la propuesta que se va a plantear, con el empleo de las herramientas de programación basad en bloques.

4.2.Marco contextual

El contexto para plantear la propuesta como un estudio de caso en el cual la estrategia ingenieril está relacionada con el empleo de herramientas para la programación basada en bloques, siguiendo las tendencias de la ingeniería de método hacia los enfoques ágiles, en la Facultad de Ingeniería Electrónica y Telecomunicaciones (FIET) de la Universidad del Cauca, en Popayán, en el sector de Tulcán. Ver figura 1.

Figura 1. Campus Tulcán, Universidad del Cauca

Fuente. http://www.unicauca.edu.co/fiet/acerca-de-la-facultad

La FIET, ha evolucionado en su oferta académica de pregrado y posgrado, siendo el punto de partida sus inicios como pionera de la formación de ingenieros de Telecomunicaciones, en la

década de los años 1960, y posteriormente, con la creación del programa de Automática y de Sistemas, en el año 1998, con asesoría de profesionales de Cuba, enriqueciendo su acción transformadora, generando profesionales con nuevas perspectivas y competencias acordes con la sociedad del conocimiento. Por eso, el programa de Ingeniería de Sistemas fomenta grupos como el de IDIS, cuya meta es la investigación y desarrollo en ingeniería del software, creado con los auspicios de Colciencias, para el avance en ingeniería del software, alcanzando reconocimiento en Latinoamérica, al situarse entre los 10 mejores grupos a nivel regional.

Así, desde el accionar de este grupo se pretende lograr mejoras significativas en el aprendizaje de los conceptos básicos de la ingeniería del software en diversos entornos educativos, desde los de pregrado en la propia institución, hasta partes interesadas fuera de ella, pero dentro del sistema educativo, particularmente los alumnos de primaria y por ello se han realizado experiencias de grupo de casos con herramientas de programación visual basadas en bloques.

Al respecto, se debe mencionar que, con referencia a los planes de área en Informática y Tecnología, las directrices esenciales están estipuladas en la guía número 30 del Ministerio de Educación Nacional (MEN), y las tendencias globales apuntan a la implementación de la programación basada en bloques combinada con la robótica, para los cursos de primaria (K-5) o para la primaria y la secundaria (K-12) siguiendo el esquema sajón de educación previa a la educación superior; esto implica el uso de herramientas como Scratch y de entornos como Mindstorms EV3.

Se busca que, las experiencias con el grupo IDIS y las metodologías allí generadas promuevan, en el futuro, una sólida fundamentación del pensamiento computacional en los estudiantes tanto de primaria como de secundaria en las instituciones públicas y privadas de la ciudad.

5. Metodología

Para el desarrollo del presente trabajo de grado se seguirá la metodología de estudio de caso adaptado a las necesidades de este proyecto de investigación, ChildPrograming como enfoque metodológico. A continuación, se describe la metodología como se desarrolló el proyecto:

5.1. Revisión de literatura

(Silamani & Guirao, 2015) conciben la revisión de literatura como un paso previo a cualquier investigación y tiene por objetivo mirar hasta dónde va el conocimiento sobre la temática que se está investigando y que aspectos de este sirven para nutrir nuestra propia investigación de manera que se puede desarrollar una sinopsis de diferentes investigaciones y artículos, de los cuales, desde una postura crítica, se pueden tomar elementos que son el punto de partida para el desarrollo de la metodología a utilizar en nuestro propio proceso de generación de conocimiento.

Por ello, la revisión bibliográfica es un proceso que se considera detallado, selectivo y crítico y que sirve para la actualización académica, la aplicación del conocimiento científico existente a una situación práctica en particular y la identificación de las aproximaciones teóricas sobre la temática particular de la que se está tratando.

En el presente caso se trate en una revisión sistema de la literatura en la que existe un protocolo bien definido para identificar, evaluar y sintetizar los referentes bibliográficos, a fin que se reduzcan los sesgos implicados en la búsqueda exhaustiva y que se responda a la pregunta de investigación, construyendo cadenas de búsqueda con palabras claves que permitan restringir los resultados de acuerdo a determinados criterios de búsqueda, de inclusión y exclusión; se examinan en bases específicas de datos artículos científicos, libros especializados y algunas tesis.

De igual manera, se seleccionaron ocho palabras claves, referentes al pensamiento computacional la programación basada en bloques y sus relaciones con la abstracción, la resolución de problemas y la colaboración, mediadas por diseños software como aquellos implícitos en el lenguaje de programación Scratch que promueve la metodología ChildProgramming.

Así mismo, se enfatizó en documentos en lengua inglesa y en los más recientes en lengua española, abarcando un rango fundamental de los últimos cinco años, utilizando para la búsqueda de la información bases de datos reconocidas como Google Scholar, Researchgate, Springer, SemanticScholar, Springer y Scielo, utilizando para las labores bibliográficas de referenciación el software de Elsevier denominado Mendeley Desktop, que permiten poseer una base de datos

actualizable de los documentos que se van a utilizar y es versátil en cuanto al formato de la referenciación, que en este caso se ha realizado con las normas APA séptima edición.

Asimismo, desde la ruta metodológica, se tiene una revisión sistemática basada en la bola de nieve, estrategia que permite establecer el estado de la ingeniería de software, mediante la síntesis de los aportes de diferentes estudios, involucrados en el ejercicio investigativo que se está adelantando; se tiene entonces un procedimiento que empieza por establecer palabras claves y cadena de búsqueda para utilizar en las bases de datos, iniciando con documentos específicos relevantes se pueden encontrar Google académico. A partir de ellos y de las referencias bibliográficas de interés en estos documentos, se puede extender la búsqueda y enfatizar en editoriales específicas; de esta manera de forma iterativa se pueden ir clasificando documentos relevantes, dejando los más antiguos en un grupo y avanzando hacia la literatura más reciente (Wohlin, 2014).
Entonces, con base en lo antes expuesto y con los aspectos característicos establecidos para una revisión sistemática de la literatura, se puede sintetizar el protocolo de revisión utilizado en esta investigación, mediante los siguientes pasos:

- Establece el objetivo el objetivo de la revisión y el intervalo de tiempo de la búsqueda.
- Identificar las categorías de análisis.
- Plantear las palabras clave para cada una de las categorías establecidas.
- Seleccionar los buscadores y las bases de datos.
- Realizar la búsqueda.
- Analizar las fuentes y desechar aquellas que no cumplan con los criterios establecidos para la búsqueda.
- Sistematizar la bibliografía en un software especializado.
- Utilizar la información recopilada dentro de la investigación.

Entonces, el objetivo de la revisión de la literatura sistemática fue la caracterización de las estrategias para el diseño software utilizando entornos basados en bloques y empleados para la enseñanza de los fundamentos del pensamiento computacional a estudiantes de básica y media, enfatizando tanto en los elementos involucrados en los procesos como en los métodos empleados en estas distintas experiencias, enfocándose en el desarrollo de la abstracción y las destrezas para la resolución de problemas.

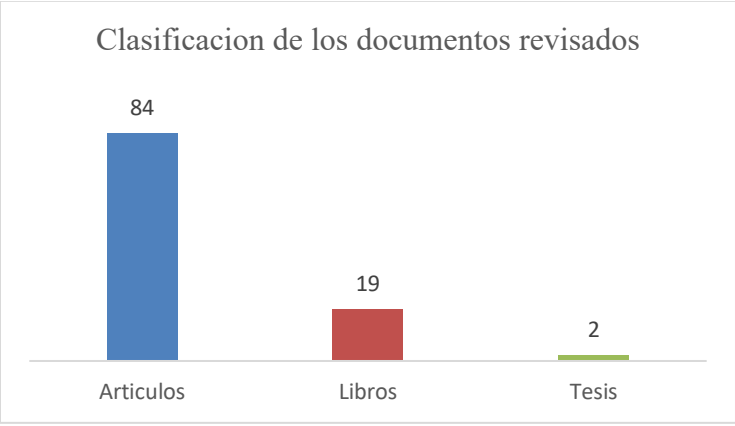
En primera instancia, se exploraron las tesis existentes sobre programación basada en bloques disponibles en línea en el enlace https://dspace.mit.edu/handle/1721.1/7582, al igual que aquellas relacionadas con el desarrollo de la programación visual utilizando Scratch. Igualmente, se examinaron los libros especializados de Springer, especialmente aquellos relativos a la interacción humano y computadora, donde se encontró una valiosa recopilación de artículos, particularmente aquellos que versan sobre el ChildProgramming.

Después, se procedió a establecer las palabras claves y las cadenas de búsqueda, entre las cuales se pueden citar: "programación basada en bloques" *and "*abstracción"; "pensamiento computacional "*and* "resolución de problemas"; "programación basada en bloques" *and* "Scratch";" ChildProgramming" *and*" colaboración"; "diseño software" *and* "pensamiento computacional";" diseño software "*and* "resolución de problemas".

Finalmente, con los resultados de la primera fase, se nutrieron las búsquedas hacia adelante y hacia atrás relativas al método bola de nieve, encontrándose con tendencias que involucraban cadenas como programación basada en bloques con metodologías ágiles.

La revisión de la literatura implicó 105 documentos recuperados de internet, en buscadores especializados como Google Scholar, Researchgate, Scielo, SemanticScholar y Springer. En la figura 4 se muestra que la mayor cantidad de fuentes corresponden a artículos (84), seguidos por libros especializados (19) y solamente 2 tesis.

Figura 2. Clasificación de las fuentes documentales revisadas

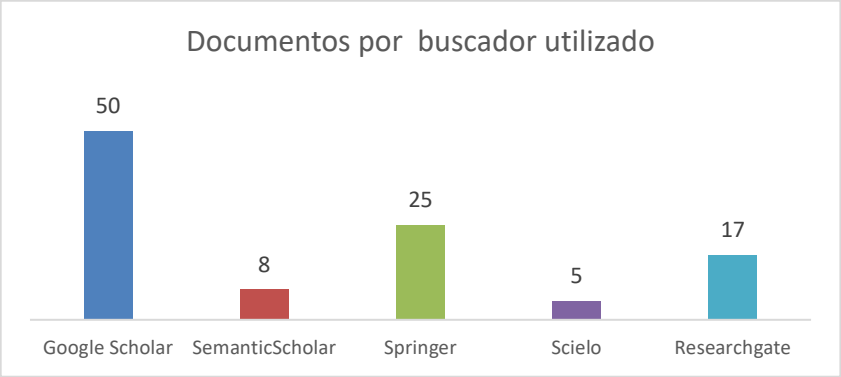


Fuente. Propia de la investigación

En cuanto a las bases de datos o buscadores de documentos, el más importante es Google Scholar

con 50 documentos, seguido por Springer, con 25 fuentes, Researchgate con 17, SemanticScholar con 7 y Scielo con 5. Ver figura 5.

Figura 3.Buscadores utilizados en la revisión bibliográfica



Fuente. Propia de la investigación

En cuanto a la selección de documentos para exponer el trabajo a nivel conceptual, como es la búsqueda de antecedentes, el marco teórico y elementos para la discusión y apoyo de las estrategias propuestas, se utilizaron palabras claves referentes al pensamiento computacional, la abstracción, resolución de problemas, y la programación basada en bloques utilizando la herramienta de software de código abierto Scratch. Ver la tabla 1.

Tabla 1. Palabras clave utilizadas en la revisión de literatura

No	Palabras claves
1	Pensamiento computacional
2	Abstracción
3	Programación basada en bloques
4	Resolución de problemas
5	Diseño software
6	Scratch
7	Child Programming
8	Colaboración

Fuente. Propia de la investigación

Con base en las palabras claves utilizadas para articular las temáticas de la revisión sistemática de literatura realizada, se pueden examinar las investigaciones que marcan la pauta en cuanto al pensamiento computacional y sus elementos, así como los métodos utilizados en diferentes experiencias, particularmente con niños y con énfasis en el desarrollo de la abstracción y la resolución de problemas.

En la primera fase de búsqueda se seleccionaron los documentos que indicaban la manera en que la programación basada en bloques impacta las destrezas computacionales, particularmente la resolución de problemas y la abstracción, a través de herramientas diversas como mBlock y Scratch y con estrategias lúdicas; se recopilan las ideas sobre pensamiento computacional y se plantean las métricas que comparan las diversas experiencias donde se enseñan los conceptos básicos de programación mediante la programación basada en bloques.

En la segunda fase, se buscan los documentos que hablan de estrategias para enseñar los conceptos básicos de la computación, particularmente las metodologías ágiles y la filosofía del pensamiento computacional, así como la ventaja de la programación visual basada en bloques, sobre las estrategias tipo texto, configurando los principios básicos del ChildProgramming y la programación basada en bloques, con entornos como Scratch.

En la tercera fase, se recopilan los documentos que permiten soportar un estudio de caso basado en las ideas de programación basada en bloques en un entorno visual con estrategias lúdicas y la filosofía ágil, promoviendo el trabajo en equipo y la abstracción.

5.2. Método

Con referencia al método de diseños software, tener en cuenta la actualidad la ingeniería de software busca nuevas alternativas al diseño tradicional y una de ellas es, sin lugar a dudas, el método ágil, mediante el cual es posible estar en comunicación continua con el usuario y establecer desarrollos que satisfagan completamente sus expectativas, optimizando el tiempo y los recursos, sean estos humanos, tecnológicos o del entorno, estableciéndose relaciones dinámicas en todo en las etapas del ciclo de vida de un proyecto de sistemas, desde la recopilación de los requerimientos, el análisis, las limitaciones, especificación, diseño y arquitectura, programación, pruebas de software, implementación, documentación y finalmente el mantenimiento (Maida & Pacienzia, 2015).

Por ello, es necesario elegir entre la opción más idónea de aquellas disponibles dentro del método ágil, ya que, de esto depende de la estrategia global que se seleccione para establecer las distintas fases y sus tareas respectivas en el proceso de diseño de software y en este caso se tiene una aproximación entre Scrum y Lean Development.

El primero porque se está haciendo énfasis sobre las buenas prácticas en el trabajo colaborativo, en

el trabajo en equipo para lograr el mejor resultado posible y en el desarrollo incremental; el segundo enfoque se selecciona porque está inspirado en las buenas prácticas que eran parte de la filosofía de la empresa Toyota japonesa de la mano o del justo a tiempo, la maximización del desempeño, la minimización de los tiempos muertos y los desperdicios.

Así las cosas, desde la filosofía de la ingeniería de método se ensamblan las ventajas de todas estas estrategias ágiles con la programación basada en bloques y los entornos de enseñanza y aprendizaje en los ambientes educativos, promoviendo la idea que los usuarios finales, son los estudiantes y particularmente los jóvenes, y que puedan escribir código a partir de resolución de problemas que estén diseñados para desplegar el pensamiento computacional.

De modo que, se puede hablar de métodos de diseño software que privilegia el aprendizaje de los principios básicos de la ingeniería del software por los usuarios finales partiendo del análisis de los requerimientos y de la creación de abstracciones, la utilización de mapas mentales, metáforas, asunción de roles mediados por la gamificación, el diseño simple, la prueba de los programas realizados y una continua interacción del estudiante con sus pares (Fronza et al., 2019).

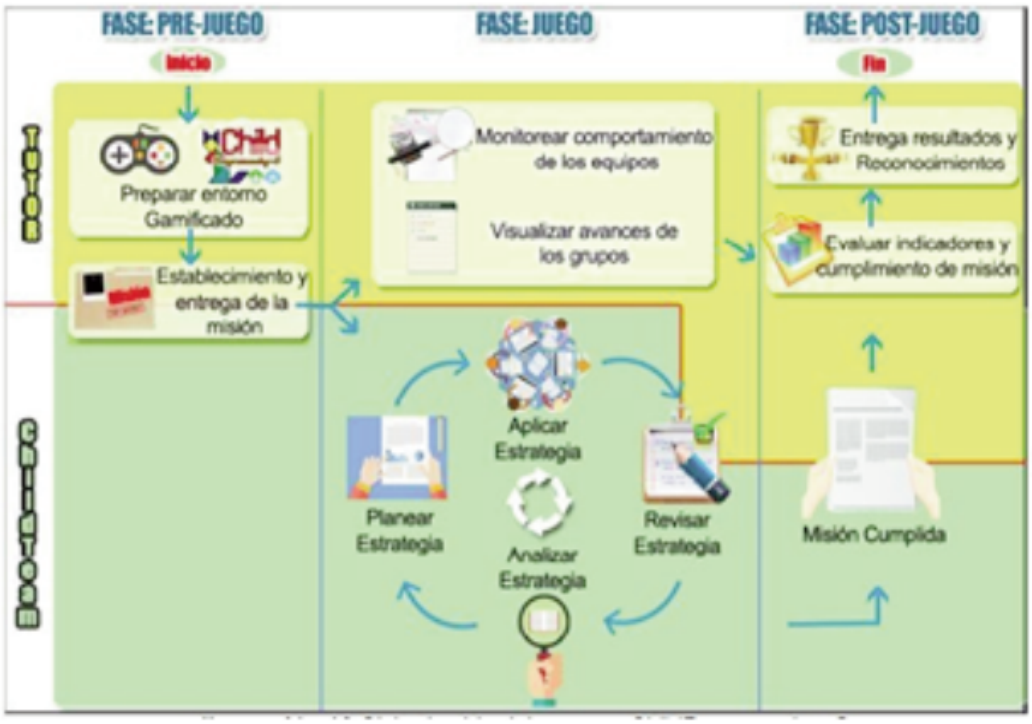
Entonces, es conveniente desde esta filosofía de método, profundizar en un enfoque metodológico que reúne todas estas características y que se denomina la metodología ChildProgramming, que se discute en el apartado siguiente.

5.3. Metodología ChildProgramming

Este trabajo sigue los pasos de la metodología ChildProgramming, desde las experiencias realizadas por (Orejuela et al., 2013; Hurtado et al., 2011; Hurtado et al., 2017) y la metodología de estudio de caso en las investigaciones de ingeniería de software, desde una perspectiva descriptiva como lo plantean (Runeson & Höst, 2009).

En este orden de ideas, se retoman las fases expuestas Hurtado (2017) para gamificar sus experiencias de la programación para niños, con énfasis en la agilidad y la colaboración y también para colocar las bases para una memoria transactiva, que comprenden una fase de pre juego, una de juego y otra de post juego. Ver figura 2.

Figura 4. Ciclo de vida de Child Programming- G

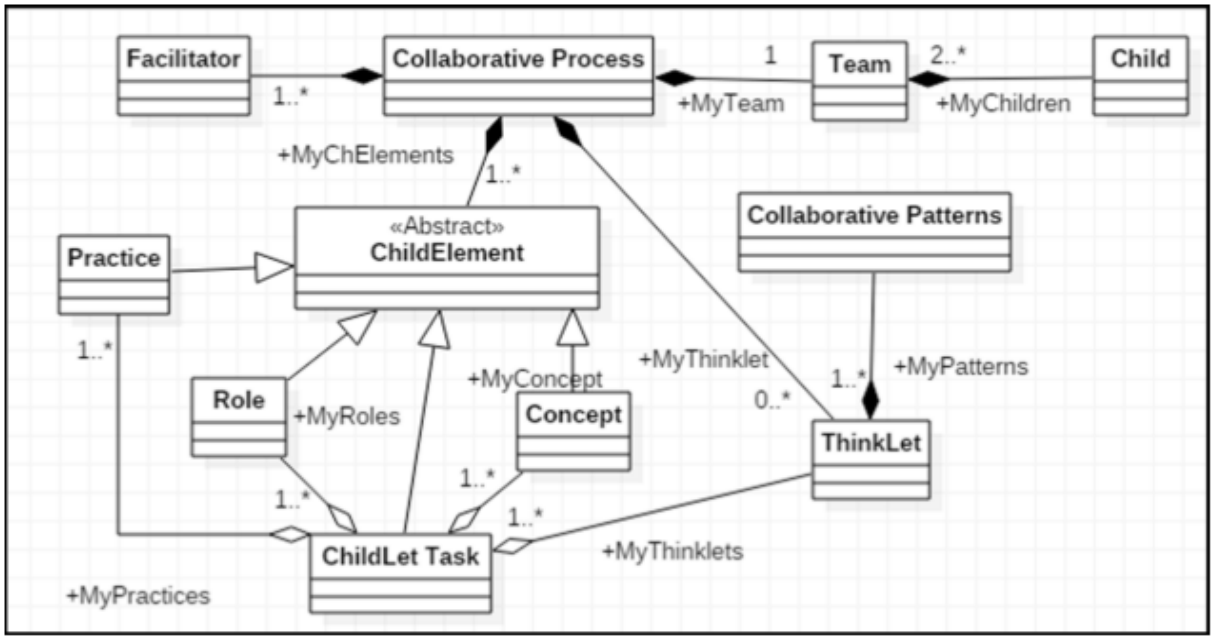


Fuente. Hurtado et al. (2017), p.101.

Este enfoque implica acciones específicas por cada fase, como se describe a continuación: En la primera fase se establece la estrategia de gamificación y los objetivos o misión de las practicas a realizar; la segunda etapa implica gestionar los equipos de trabajo y mirar sus avances en cuanto a las tareas grupales de programación; finalmente, se tienen las fases de evaluación de los procesos o destrezas adquiridas y de la entrega de los resultados.

Esto conlleva a tener en cuenta la arquitectura que subyace en ChildProgramming, es decir, los conceptos, prácticas, componentes colaborativos (thinklets), cognitivos y ágiles, así como el objetivo fundamental de éste método y su evolución, desde su propuesta original en 2013, la versión G, con gamificación y la propuesta GTM, de la memoria transactiva, y el aporte final que se enfoca en la abstracción, de modo que el punto fundamental es la determinación de los mecanismos de abstracción de una muestra de 14 niños de 8 a 12 años, a través de procesos incrementales, ejemplos y modelos mentales, que permitan desplegar la descomposición, la recursión y la generalización. En la figura 3, se expone de forma simbólica la arquitectura de ChildProgramming

Figura 5. Arquitectura conceptual de ChildProgramming



Fuente. (4th Iberoamerican Workshop, 2019), p.59

En cuanto, a la parte de la metodología de investigación, el primer paso es realizar la revisión de la literatura que permita el conocimiento de los aspectos relevantes referentes a la estrategia de diseño de software en el contexto del desarrollo del pensamiento computacional en niños.

En segundo término, es proponer un método de diseño de software que permita integrar mecanismos de abstracción para facilitar a los niños la construcción, descomposición e integración de nuevos bloques considerando las estrategias caracterizadas. Esto implica definir la herramienta que se va a utilizar para las actividades de programación basada en bloques, así como los elementos que se deben enfatizar en las actividades que se van a proponer y aplicar en la prueba piloto.

Finalmente, mediante un grupo de caso específico, se evalúan los resultados y métricas propuestas para proponer en el futuro pautas para proceder al diseño de la estrategia didáctica a seguir en el diseño de software que permita integrar mecanismos de abstracción para facilitar a los niños la construcción, descomposición e integración de nuevos bloques considerando las estrategias caracterizadas.

En cuanto al estudio de caso, aplicado a las disciplinas de diseño de software y similares, en esta ocasión se utiliza con énfasis exploratorio, en el sentido de utilizar la programación basada en bloques con estudiantes de 8-12 años, de diversos entornos educativos, es decir, con diferentes

conocimientos de programación, de manera que sea posible evaluar la adquisición de las competencias de pensamiento computacional, después de 6 sesiones de 4 horas.

6. Caracterizar las estrategias de diseño de software en el contexto del desarrollo del pensamiento computacional en niños mediante revisión de la literatura y un caso exploratorio

6.1. Caracterización de las estrategias de diseño software

(Ahmed et al., 2020) teniendo en cuenta el hecho que, a nivel mundial, la programación visual basada en bloques es ampliamente utilizada para presentar los conceptos de la computación a los estudiantes de primaria y secundaria, expone la manera en que es posible la síntesis de tareas de programación visual que se van a utilizar estos entornos educativos para lograr desarrollar un nivel deseado de dificultad en los ejercicios sobre conceptos específicos de programación; promueven un nuevo método lo que genera automáticamente un conjunto nuevo de tareas y su código, a partir de una tarea visual de referencia de cada código solución de entrada, con lo cual promueven el aprendizaje de la programación

Así, mediante un árbol de búsqueda de Monte Carlo realizan una ejecución simbólica del código de salida para obtener tareas nuevas en forma de rompecabezas, que difieren de las tareas de entrada propuestas en su disposición visual, aplicando pasos específicos: primero, se formaliza el problema de la síntesis de tareas para la programación visual basada en bloques y segundo lugar, muestra la efectividad a través de una extensa evaluación empírica en la que el usuario estudia con base en tareas de referencia entran en las plataformas de programación del mundo real

Por tanto, este artículo corrobora la importancia de la programación basada en bloques y de estrategias para introducir conceptos de la informática.

(Gunbatar & Turan, 2019) describen el efecto que tiene la programación basada en bloques sobre las destrezas computacionales de estudiantes de enseñanza media en escuelas de Turquía utilizando la herramienta mBlock para procesar las instrucciones de programación computacional, abarcando los conceptos de software, problemas, algoritmos, los pasos del algoritmo que pertenecen a un problema, formas geométricas y obtuvieron diferencias significativas en los estudiantes incrementando el pensamiento computacional y transformando la introducción a la programación, resultados a los que se llegó mediante la aplicación de la Escala de Niveles de Pensamiento Computacional (CTLS). Las sesiones de enseñanza que hicieron con una intensidad de dos horas semanales durante un lapso de 12 semanas con 82 estudiantes de sexto grado de una escuela pública de la provincia de Van.

La utilización de mBlock obedece a que esta herramienta de código abierto permite la programación de robots basados en Arduino o Mbot sin necesidad de conexión por cable y promueve el mejoramiento de las destrezas de resolución de problemas, lo que ha sido por los resultados, que indican un mejoramiento de las habilidades de pensamiento crítico y por tanto, confirma que la programación basada en bloques mejora el aprendizaje de los paradigmas de programación y el pensamiento computacional con los que se desarrolla el proyecto de interés.

(Serna & Polo, 2014) exponen el papel fundamental de la abstracción para remover los elementos no fundamentales y luego proceder a las generalizaciones, enfatizando los conceptos clave, el núcleo común, en diferente situación, contribuyendo así a que los estudiantes de ingeniería de sistemas y de software mejoren sus habilidades para analizar, conceptualizar, modelar y resolver problemas referentes a la programación, la arquitectura de sistemas de software o de los sistemas de información y enfatiza en la necesidad que, en los programas de enseñanza de la informática, a todos los niveles, se den pautas para que los estudiantes desarrollen sus habilidades de abstracción, para enfrentar con éxito los retos en la resolución de problemas.

En este orden de ideas, este trabajo de grado propone actividades de diseño que faciliten la abstracción y la representación de las soluciones propuestas a cada uno de los problemas planteados con escolares de una institución.

(Roscoe et al., 2014) exponen la forma en que se puede enseñar el pensamiento computacional mediante estrategias lúdicas y construcción de robots, lo que exige que los estudiantes aprendan sobre el código software, planteando retos relacionados con la escritura de su código fuente, implicando la adquisición de nuevas destrezas en pensamiento computacional, y por esta necesidad se promueven diversas destrezas computacionales desde la enseñanza en las escuelas secundarias, mediante tres estrategias fundamentales: Printcraft, para adquirir competencias sobre diseño asistido por ordenador y manufactura aditiva a través del diseño y construcción de mini mundos con la herramienta abierta Minecraft e impresoras 3D; la adquisición de una nueva perspectiva sobre la tecnología existente, mediante una aplicación Android como AppInventor, que es un entorno grafico de programación; la creación, en equipo colaborativo, de robots diversos con Arduino y Scratch.

Con estas actividades se promueve la percepción de la importancia de la computación y del pensamiento computacional, como técnica de resolución de problemas de aplicación amplia en

diversos campos del saber humano, y se enfatizan los componentes claves de esta técnica: abstracción, análisis lógico, pensamiento algorítmico, eficiencia e innovación. Así, se motiva y empodera a los estudiantes para que mejoren su pensamiento lógico y competencias de resolución de problemas, especialmente en los grados escolares básicos y en edades entre los 9-11 años.

(Hermans & Aivaloglou, 2017) introducen los conceptos básicos de ingeniería de software con niños entre los 7 y 11 años de edad, con ayuda de un curso de programación en Scratch que permita introducir dichos conceptos, tales como problemas de depuración, calidad de código, refactorización, entre otros. Los resultados muestran que no hay diferencia en las puntuaciones de los estudiantes entre los conceptos de programación y los conceptos de ingenieríade software, también sugieren que sí es posible enseñar estos conceptos para este grupo de edad. Este trabajo da el aval para dar inicio a un modelo que, basado en la capacidad de abstracción de los niños, les permita obtener un mejor desempeño en informática.

(Reyes & Saavedra, 2016) quienes proponen una estrategia en la que evalúan un modelo colaborativo y lúdico que facilite el aprendizaje de la construcción de software en equipos de desarrollo conformados por jóvenes.

Los anteriores trabajos remiten, en primer lugar, a las dificultades que tienen las personas, particularmente los niños y novicios en el mundo de la programación cuando deben aprender a programar, puesto que existen concepciones erróneas que provienen de los primeros niveles educativos y que se relacionan con *los bucles o ciclos de la programación* estructurada y que tienen repercusiones en los niveles universitarios, de manera que es necesario buscar métodos pedagógicos que subsanen estas malas concepciones mediante el empleo de diferentes lenguajes, es decir, métodos de diseño software, como por ejemplo *Scratch*, Logo y Python; en esta búsqueda, es necesario tener en cuenta la teoría del desarrollo cognoscitivo planteada por Jean Piaget porque tiene etapas muy específicas: de los siete a 11 años, se tiene la etapa de las operaciones concretas, pero en el pensamiento abstracto, empieza adquirirse a partir de los 12 años y esto implica concentrarse en estudiantes de educación cuando se quiera enfatizar en la parte del pensamiento computacional que tiene que ver con la abstracción y la resolución de problemas (Mladenović et al., 2018).

Esto impacta también la estrategia de enseñanza de la programación y la herramienta utilizada para enseñar los rudimentos del diseño de software abarcando las métricas y direccionamientos de las

aplicaciones posteriores de este conocimiento, fundamentalmente en entornos colaborativos y en campos diversos del saber humano entre los que pueden contarse las matemáticas, en el lenguaje, la robótica o la biología, es decir, la formación en competencias STEM (ciencia, tecnología ingeniería y matemáticas), donde lenguajes visuales basados en bloques como Scratch son muy utilizados debido a que permiten llevar a cabo actividades basadas en juegos, es decir la utilización de estrategias lúdicas para enseñar las pautas de la secuenciación, de una correcta implementación de un sitio simple o de ciclos anidados, que son bloques de construcción básicos para una formación sólida en las disciplinas STEM, permitiendo el fortalecimiento del denominado *pensamiento computacional*, entendido como el conjunto de estrategias, ideas y procedimientos que las personas adquieren a través de su accionar en el diseño de programas, software y hardware para computadoras, lo que implica que "el pensamiento computacional es esa manera de pensar generada mediante la comprensión de los computadores y las tecnologías de información" (Abelson & Kong, 2018).

El pensamiento computacional, como proceso de resolución de problemas, se caracteriza por: formulación del problema, organización lógica y análisis de los datos, estando la información representada mediante abstracciones, para facilitar el encuentro de soluciones automatizadas, óptimas y eficientes, mediante algoritmos, que pueden aplicarse en diferentes situaciones; va de la mano con destrezas como la persistencia para enfrentarse a problemas difíciles, complejos, con grados diversos de ambigüedad que exige no solamente la búsqueda de soluciones desde la perspectiva de mejoramiento continuo sino también el *trabajo en equipo;* exige diferentes enfoques metodológicos como micro mundos, las transformaciones de los entornos visuales a texto y viceversa, la computación basada en objetos y el establecimiento de escalas que permitan medir las diversas destrezas desarrolladas por los estudiantes cuando, en diversos entornos educativos y con distintos enfoques metodológicos y herramientas, intentan adquirir competencias de este tipo de pensamiento; estas *métricas*, en función del nivel escolar y el desarrollo cognoscitivo de los estudiantes, permiten evaluar si tienen unas destrezas mínimas de pensamiento computacional (Abelson & Kong, 2018).

Las métricas aplicables en este caso son las de los niveles 1.B, correspondientes a las edades de 8-11 años, es decir, estudiantes de grados 3-5 de básica primaria, con tópicos relativos a cinco categorías: sistemas de computación, redes e internet, análisis y datos, algoritmos y computación,

impactos de la computación(CSTA, 2017).

De estas, son de interés para el estudio de caso que se lleva a cabo las relativas a la comprensión y empleo de los *pasos básicos para la resolución algorítmica de problemas*, es decir, la formulación y exploración del problema, el examen de instancias de muestras, diseño, implementación y prueba, al igual que aquellas relacionadas con el desarrollo de una *comprensión simple de un algoritmo dado*, es decir, la búsqueda, secuencia de eventos, o clasificación, empleando ejercicios de computación libre; también, la exploración de potenciales soluciones a problemas software mediante *estrategias de resolución de problemas específicas*, la manera en que se emplean las herramientas visuales para presentar u organizar la información, enfatizando en las relaciones y su estructuración para llegar a un fin (Abelson & Kong, 2018).

De igual forma, aquellas relativas a los algoritmos y la programación, como son: *la modularidad* (descomponer programas complejos en partes más simples, reutilizar parte de código); el desarrollo del programa (tomar en cuenta las ideas expuestas por otros para la planeación y ejecución de un programa, realizar *la depuración lógica*, asumir roles dentro de un entorno colaborativo); discutir la manera en que la tecnología y las herramientas utilizadas impacta en el proceso de desarrollo de software (CSTA, 2017).

Así mismo, la destreza para corregir errores lógicos (bugs) es fundamental para aprender a programar, particularmente en los niveles superiores, donde la abstracción se desarrolla en la enseñanza del pensamiento computacional; sin embargo, dada la amplia utilización de la programación basada en bloques en los niveles iniciales es importante incluir recursos accesibles adaptables que permitan ayudar a que estos estudiantes comiencen el desarrollo de esta destreza, lo que se facilita empleando Scratch, por lo que es necesario enfatizar en la caja de herramientas que posee esta plataforma visual de programación para las *labores de depuración lógica* (debugging) (Wang, 2021).

Por lo tanto, la fluencia digital, la búsqueda constante de ideas y la motivación con referencia a la programación, particularmente de los niños, implica colocar al alcance de todos un entorno o plataforma que permita que todos aprendan a programar sus relatos interactivos, juegos, animaciones y simulaciones y compartirlas con otros, dando realidad a la fluencia digital que engloba el diseño, creación y reutilización del código, la búsqueda, la interacción con otros, y el trabajo colaborativo creativo, propio de la web 2.0 y la galaxia Internet. Por ello se creó *Scratch*

desde el software libre y con la filosofía de la programación basada en bloques, y la denominada *espiral del aprendizaje creativo: imaginar, crear, jugar, compartir, reflexionar, imaginar.* Esto se traduce en idea poderosas como trabajar en proyectos, de forma colaborativa, a través del juego, compartir y reflexionar creativamente(Resnick et al., 2009)(Dhariwal, 2018).

Así, el pensamiento computacional refiere a procesos que comprenden la formulación de problemas en una forma que sus soluciones puedan abordarse mediante pasos específicos de computación, lo que permite automatizar su implementación mediante algoritmos; la elección de los modelos computacionales(modelos matemáticos abstractos) adecuados para la formulación de problemas, y de estrategias de diseño para llegar a algoritmos óptimos, y poder ser utilizados con las plataformas pertinentes, que en este caso remiten a la programación basada en bloques en entornos visuales como Scratch, que permiten involucrar a cualquier persona en la programación, desde el entorno familiar, con asistencia de los padres o, en las escuelas, en los grados de básica primaria o en la secundaria, privilegiando las comunidades de creadores, el trabajo en equipo, creando inteligencia colectiva a través de talleres ((Aho, 2012);(Ricarose, 2016)

6.2. Constructos para un estudio de caso

Dado que el pensamiento computacional es una forma específica de resolución de problemas que implica la abstracción, es importante explorar los conceptos que soportan la metodología ChildProgramming -A por qué remite a modelos mentales compartidos implicados en frentes niveles de abstracción utilizados por los niños a lo largo de las diferentes etapas que componen este entorno metodológico: pre- juego, juego y pos - juego. La primera fase abarca la conformación de los grupos de trabajo y la entrega de la misión que deben llevar a cabo los niños, con lo cual se pueden distribuir tareas dentro de los grupos. En la etapa de juego, se aplican las iteraciones o rondas mediante las cuales los grupos de trabajo coordinar esfuerzos con el objetivo de cumplir la misión encomendada; se tiene en tareas específicas como la planificación de la estrategia, su aplicación, su revisión y finalmente el análisis de los resultados obtenidos con ella, lo cual concuerda con las pautas de resolución de problemas dadas por George Polyá.

Por esta razón la planificación implica definir tareas para arrancar con un diseño inicial de solución; la aplicación tiene como objetivo generar métricas de desempeño colaborativo de los integrantes del equipo; la revisión es el momento de verificación, por parte del docente, de los avances realizados por cada equipo y genera recomendaciones o sugerencias que deben ser puestas en

práctica a partir de una nueva iteración ; en la etapa de reflexión todos los integrantes de los equipos de trabajo evalúan sus aportes, colaboraciones y compromiso y se deben aproximar de forma objetiva la manera en que se está llevando a cabo la solución del problema planteado.

En la fase de pos-juego, se entrega el resultado de la misión cumplida al docente quien realiza la evaluación de la solución alcanzada a la luz de los objetivos de aprendizaje planteados en función de ciertas características presentes en la literatura como son los aspectos cognitivos, colaborativos y las prácticas ágiles, que permiten concentrarse en los elementos importantes, desechar las inútiles y maximizar los resultados, al permitir el alineamiento estratégico de los elementos aportados por ChildProgramming como son roles, los conceptos y las prácticas, que son de suma importancia en cada uno de los componentes de esta metodología: colaborativo, que engloba a coger la filosofía el trabajo en equipo; cognitivo, implica aplicar los pasos de la resolución de problemas y las reglas de juego, y los valores de corresponsabilidad; ágil, que implica adoptar la filosofía del mejoramiento con, de la simplicidad diseño que permita la realización de tareas complejas.

El enfoque metodológico ChildProgramming puede representarse gráficamente en sus distintas fases y con énfasis en sus distintos elementos y componentes.

De igual manera, dado que en el enfoque ChildProgramming el componente colaborativo es fundamental para llevar a cabo las misiones y soportar los procesos o niveles de abstracción, sus elementos están mediados por las clases roles, conceptos y prácticas, remite a los algoritmos, la lógica, la descripción de eventos, la recursión, la descomposición, el ocultamiento y el despliegue funcional, con relación a objetos, en la medida en que se está en un entorno de programación visual basado en bloques, de manera que sus elementos se relacionan con estos mecanismos de abstracción.

Así las cosas, para evaluar los elementos de ChildProgramming- A, como son la abstracción, el conocimiento compartido, la incrementa habilidad, que devienen en el pensamiento computacional, es importante diseñar un estudio exploratorio, con el propósito de aproximarse a la abstracción en la cotidianidad del niño y la forma en que conoce el entorno de programación Scratch y sus elementos; esto implica empezar con la exploración de los pasos de la resolución de problemas, la representación mediante gráficas, esquemas, listas u otros elementos visuales de las situaciones referentes al problema particular; de allí, explorar el entorno de Scratch y sus elementos, con ayuda del docente, enfatizando que este entorno de programación visual, es una manera de construir y

representar soluciones a problemas; para ello se utilizan ejemplos específicos, paso a paso como tutoriales sobre las huellas, los movimientos, de un gato; luego, se dejan retos como el dibujo de figuras específicas en el entorno Scratch.

En estes orden de ideas, mediante listas de verificación se conoce si los estudiantes han retenido los elementos y funcionalidades del entorno Scratch, preguntando el nombre del elemento y su propósito. De igual forma, se requiere que los estudiantes hagan un reconocimiento de las estructuras de repetición y de su empleo en el trazado de polígonos regulares, conocimientos que se van a llevar luego, en la práctica, al entorno Scratch.

A partir de las respuestas obtenidas mediante los instrumentos, como las listas de verificación, se evalúa el pensamiento computacional y la abstracción requerida para dar solución a la generación de los polígonos, que en realidad se constituyen en puzles, que pueden ser automatizados. Se tiene entonces un proceso incremental que va del problema inicial, el conocimiento de los elementos básicos, la identificación, la aplicación y la evaluación.

De igual forma, se puede enfatizar en los componentes abstractos que surgen en el trabajo colaborativo; se empieza con los componentes de planificación y decisión de los diferentes equipos sobre los elementos de Scratch que pueden utilizar en una situación particular. Se puede establecer una actividad de construcción de un juego consistente en un laberinto y en la manera en que los diferentes grupos clasifican en categorías su reto, y qué resultados obtienen.

Así mismo, con la estrategia lúdica se pueden establecer retos colaborativos utilizando personajes de cuentos como la sirenita, la tortuga y la liebre, Hansel y Gretel, o Caperucita Roja, refiriendo a elementos del pensamiento computacional como el paralelismo, el pensamiento lógico, control de flujo, interactividad con el usuario, representación de la información, sincronización y abstracción.

6.3. Desarrollo de un estudio retrospectivo a partir del estudio de caso ChildProgramming-A

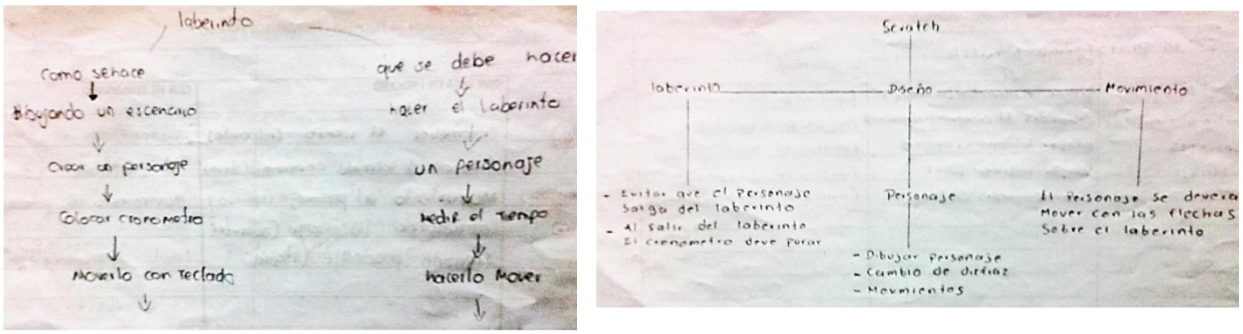
Este estudio de caso toma como base la experiencia del estudio de caso 2 reportado previamente por el grupo, y de manera retrospectiva analiza dos proyectos con el fin de evidenciar algunos constructos que podrían ser útiles para el modelo de diseño. El objetivo es explorar los elementos que los estudiantes entre 10 y 13 años usaron libremente en el espacio de diseño dentro de un contexto de trabajo en equipo de 4 o 5 estudiantes haciendo uso de modelos mentales para representar las abstracciones de la solución (espacio de diseño). Así que las unidades de análisis

son los modelos mentales obtenidos por los equipos de trabajo, su selección es por disponibilidad, pero los equipos fueron definidos aleatoriamente. El estudio de caso es de tipo embebido (dos unidades) y de tipo exploratorio. La pregunta de investigación que se realiza es ¿Cuáles son los principales constructos usados por los equipos para describir el diseño de su solución a través de un modelo mental compartido?

Para su desarrollo se realizaron 2 actividades cada una de 4 horas bajo las cuales se aplicaron algunos conceptos de los modelos mentales compartidos y los mecanismos de abstracción trabajados previamente por el grupo IDIS. Estos modelos fueron estudiados a fin de identificar algunos constructos.

Dentro de los resultados se pudieron identificar algunos elementos coincidentes con la literatura y otros presentes por la investigación previa y por la concepción de los programas en un lenguaje orientado a bloques como Scratch: La siguiente Figura muestra dos modelos mentales obtenidos del estudio de caso de Zúñiga et al, 2016 con el fin de analizar los constructos. Ver figura 6.

Figura 6.Proceso ChildProgramming



Fuente. (Zuñiga-Muñoz, 2016), p.28.

Haciendo un análisis de los modelos se pueden identificar los siguientes constructos

1. Escenario (Laberinto, muy específico en ambas unidades de análisis)
2. Personaje de manera muy genérica en ambas unidades de análisis. Es poco relevante la personalidad e imagen del personaje ya que tiene más relevancia la misión de sacarlo del laberinto.
3. Acciones asociadas al laberinto (de poner obstáculos y de terminar el juego a la salida), también una temporalidad para terminar el juego
4. Acciones asociadas al personaje, moverse a través del laberinto.
5. Jerarquías poniendo de manifiesto:

a. Jerarquía de Procesos de la interacción interviniendo escenarios, personajes y movimientos en la raíz la interacción, y como hijas
b. Jerarquía formada por la tríada escenario, personajes y acciones

En síntesis, se pueden identificar los constructos: escenario, personaje, movimiento y acción. Algunos de estos constructos se especifican sin mucha abstracción como Laberinto y Movimiento.

7. Proponer un método de diseño de software que permita integrar mecanismos de abstracción considerando las estrategias caracterizadas

Desde la revisión de la literatura, y de la caracterización realizada anteriormente, es posible afirmar que los esfuerzos realizados por la Universidad del Cauca para promover metodologías de enseñanza del pensamiento computacional y el diseño software, no sólo entre sus estudiantes de pregrado, sino también en los grados básicos de primaria y secundaria, se ven reflejados en el énfasis dado a la metodología denominada Child Programming-C, basado en la colaboración, como aporte a la metodología de desarrollo de software denominada programación para niños (Child Programming), estrategia que se ha puesto a prueba mediante experiencias desarrolladas con estudiantes en establecimientos educativos, con la finalidad de hacer que se tenga conciencia de la importancia del trabajo colaborativo en equipo para el mejoramiento de las habilidades sociales, cognitivas y del pensamiento computacional, empleando *estrategias lúdicas, la colaboración y la agilidad*; Child Programming (ChP) se convierte en un entorno ideal para que los niños aprendan a programar con base en un modelo de gestión de conocimiento propio de la sociedad actual, permitiendo a través de la gamificación, adentrarse de manera amena en los principios básicos del desarrollo del software y competencias concomitantes de ((Chimunja et al., 2017)); (Hurtado et al., 2011); (Hurtado et al., 2017), (Orejuela et al., 2013)

7.1. Propuesta de Método

Entonces, el método de diseño propuesto se basa en la metodología ChildProgramming, utilizando la plataforma de programación visual basada en bloques empotrada en Scratch, desde las directrices suministradas por diferentes guías y literatura relacionada con esta plataforma de programación, y con base en materiales disponibles en reconocidos MOOC y fundamentalmente, aquellas dadas en https://www.media.mit.edu/posts/introducing-scratch-3-0-expanding-the-creative-possibilities-of-coding/.

Así las cosas, es necesario puntualizar que el primer énfasis es el de la creatividad puesto que se busca motivar a las personas jóvenes para que a través de una computación creativa se conecten con sus intereses y valores referentes a la computación y dejen surgir lo mejor de lo que son capaces, su imaginación, sus intereses, permitiéndoles adquirir competencias en pensamiento computacional, partiendo de una codificación de los programas basados en bloques, alternativa a la programación tradicional, tomando parte activa a través del diseño de escenarios y distintos

proyectos donde interactúan con otros para crear contenidos, discutir y divertirse, con prácticas diseñadas tanto para el nivel elemental, medio o avanzado privilegiando uno de los métodos más aceptados del diseño software en la actualidad: los métodos ágiles.

Los métodos ágiles han trascendido la ingeniería de software y se han popularizado en otras ramas del saber humano, principalmente la gestión de proyectos puesto que la creación de software exige una continua interacción con los clientes para que el producto cumpla las necesidades de los clientes y pueda ajustarse de forma óptima a los retos de un entorno cambiante, de modo que, la agilidad está en función del trabajo en equipo y va de la mano con la filosofía del mejoramiento continuo, de la producción con calidad, de eliminar los desperdicios, no solo de materiales, sino de tiempo; de optimizar los procesos, de manera que se agregue valor a los productos. Los postulados fundamentales de estos métodos de acuerdo a (Gacitúa-Bustos, 2014) son los siguientes: Los individuos y sus interacciones son más importantes que los procesos y herramientas; un software funcional importa más que la documentación que lo soporta; la negociación con los clientes tiene mayor prioridad que los pactos de negocios y la adaptabilidad al cambio, la flexibilidad, tienen mayor peso que seguir un plan.

De hecho, la agilidad se ha convertido en parte de los estándares sobre la gestión de proyectos, complementando el modelo de software, como estrategia conceptual y metodológica para entregar un producto específico que siga los principios agiles, las visiones de desarrollo adaptativas, donde no solo son fundamentales las iteraciones sino también el mejoramiento continuo y la optimización de los recursos, con base en la minimización de desperdicios y la creación de valor agregado, en un entorno de cambios muy rápidos, que exigen la simplificación de los procesos, resultados tangibles y división de grandes proyectos en pequeñas piezas, por lo que se enfatiza en la gestión de grupos de trabajo (PMI, 2021)(Marion & Richardson, 2023).

Así, la filosofía del pensamiento computacional, con base en la programación visual basada en bloques, que permite la implementación del trabajo en equipo, del empleo de la lúdica para motivar a niños y jóvenes a programar, para la creación de contenidos, de juegos, de programas orientados hacia la Internet de las cosas, pensados para la robótica, o las aplicaciones con sensores, con Arduino o Raspberry pi, y con el empleo de herramientas libres como Scratch ,Tynker o Blockly , y aún, de App Inventor, pensada para las plataformas móviles, o con entornos como Lego Mind Storms EV3, es la que está en concordancia con la sociedad de la información y la cuarta revolución

industrial, implicando que la programación visual basada en bloques debe ser aplicada y evaluada en diferentes entornos educativos, desde diferentes aspectos, como el precio, las capacidades 2D y 3D,el precio, los grupos objetivo de usuarios, los tipos de proyectos que se pueden desarrollar, el sistema operativo, las capacidades de emulación o la simulación (Zamin et al., 2018)(Hu et al., 2021).

De acuerdo a las investigaciones de Hu et al. (2021) Scratch es un software basado en bloques, es decir, que privilegia las funciones a través de operaciones visuales de arrastrar y soltar, empleando rompecabezas, sin utilizar un generador de código, que además de ser libre, es muy simple de utilizar, con una interfaz de usuario 2D que se despliega en una sola pantalla, que puede utilizarse sobre Windows, Mac Os o Linux, en sus diversas distribuciones, con una alta usabilidad, permitiendo la creación de cuentos, animaciones y juegos y que es empleado por estudiantes de 8-16 años, aunque puede ser empleado por cualquier persona; por estas características es preferido sobre Alice o App Inventor, si bien lo más importante es que los logros de los estudiantes son superiores. De hecho, estos autores sostienen que:

> El uso de la herramienta de programación visual basada en bloques posee una significativa ventaja sobre la efectividad del aprendizaje de la programación con referencia a los logros académicos de los estudiantes. De acuerdo a los resultados del análisis, Scratch es más efectivo en impulsar logros académicos en relación al aprendizaje de la programación; esto refleja el hecho que los estudiantes universitarios consideran que la herramienta más apropiada y poderosa para la introducción a la programación en todas las edades del sistema educativo, es, Scratch.(Hu et al., 2021)

Por otra parte, Scratch permite desplegar las destrezas de programación a través de la espiral del aprendizaje creativo expuesta por Resnick y que es fundamental para los diseños de simplicidad visual, agilidad y colaboración, enfatizados en ChildProgramming. Ver figura 7.

Figura 7. Espiral del aprendizaje Creativo

Fuente. Elaborada con base en Dhariwal,2018, p.6.

Entonces, la propuesta consiste en enfatizar en la resolución de problemas a través del pensamiento computacional mediante el lenguaje visual basado en bloques Scratch, utilizando los principios dados en ChildProgramming, la espiral del aprendizaje creativo y los siguientes contenidos:

- Introducción: ¿Qué es Scratch? ¿Para qué sirve? Ejemplos de juegos con Scratch, instalación, mi primer proyecto en Scratch.
- Reconocer el entorno de trabajo de Scratch
- Reconocer el entorno de trabajo del editor de pinturas
- Diseñar y crear y editar Objetos, Disfraces, Fondos; y editar Escenario, Dar instrucciones básicas a Objetos
- Movimiento de objetos y animaciones básicas con las teclas, aleatoriamente, mouse.
- Instrucciones de Bucles o repetidores - Realizar movimientos aleatoriamente, Cambiar disfraces, Cambiar fondos
- Crear historias interactivas con Scratch incorporando instrucciones como: pensar, pensar por N segundos, decir, decir por N segundos, cambiar disfraz e instrucciones de sonido.
- Crear programas que manejen eventos (sensores)
- Lápiz
- Variables y contadores - Contando acciones como las colisiones, aciertos, errores.
- Interactividad con el usuario

De igual forma, se siguieron ideas dadas en http://scratched.gse.harvard.edu/guide/ , con referencia a los contenidos prácticas en las categorías de: experimentación e iteración; prueba y depuración

lógica; reutilización y remezclado; abstracción y modularización, con los contenidos apropiados para K-3, de acuerdo a las edades de 8-12 años, es decir, que aplican los videos propios de la escuela elemental y media. Los siguientes son los enlaces web de los videos tomados como base para el desarrollo de los contenidos de la propuesta:

https://vimeo.com/106046496?embedded=true&source=vimeo_logo&owner=5170010
https://vimeo.com/106046496?embedded=true&source=vimeo_logo&owner=5170010
https://vimeo.com/106046496?embedded=true&source=vimeo_logo&owner=5170010
https://vimeo.com/106437612?embedded=true&source=vimeo_logo&owner=5170010
Y la guía utilizada para los conceptos básicos de Scratch 3.0 disponible en el enlace siguiente:
https://comunidadatenea.org/uploads/iniciativas/830/documentos/Scratch-Comunidadatenea.pdf.

7.2. Aspectos Generales del Método Block-Based Design (BBD)

El método, obtenido luego de la revisión de la literatura y del estudio exploratorio retrospectivo, es la apuesta del grupo IDIS para que el proceso ChildProgramming empiece a sistematizar los métodos y técnicas asociadas, es decir, explotar la utilidad del método BBD (Block-Based Design) para la descomposición del sistema, para la tarea propia de la etapa de ChildProgramming denominada *Planear la estrategia* en la cual se debe planificar el trabajo a partir de las ideas propias de la descomposición de la misión entregada a los niños. Ver figura 8.

En la figura se pretende ilustrar que, aunque los pasos iniciales de descomposición se aplican a la estación *Planear Estrategia*, los pasos siguientes en el proceso de refinamiento propuestos por el método de diseño apuntan más a la aplicación de la estrategia, para que su aplicación no se limite sólo a la programación como la idea original de ChildProgramming.

Figura 8. BBD incrustado en ChildProgramming

Fuente. Propia de la investigación

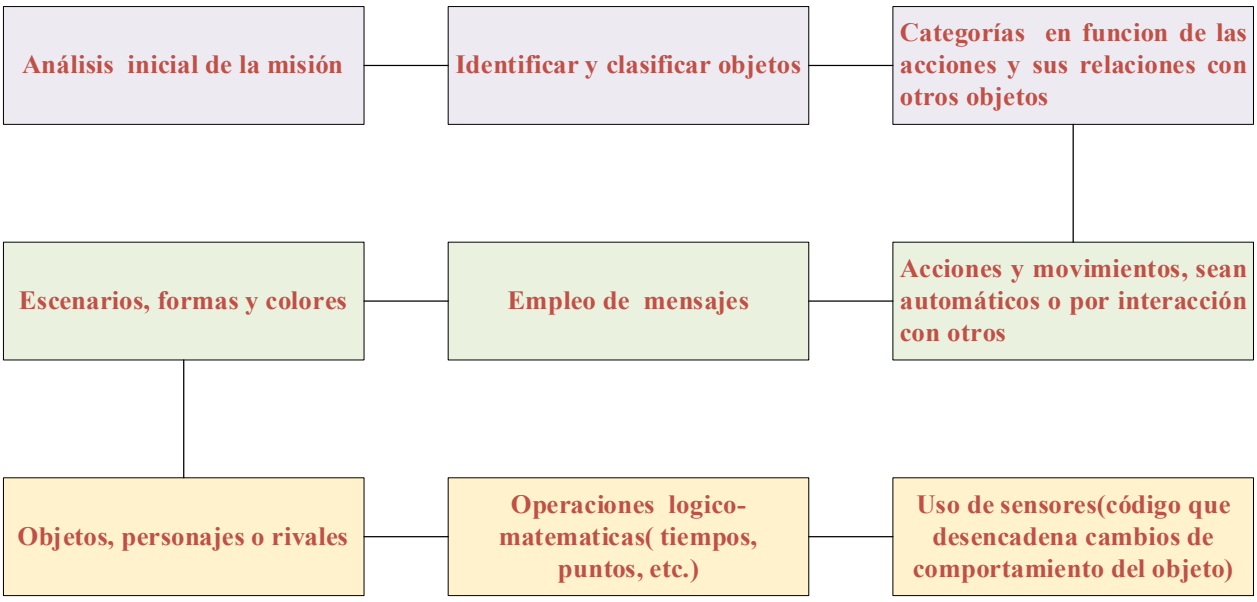
Así, desde la ingeniería de método, BBD se organiza en función de aspectos del proceso y del producto, que devienen en dos componentes: El enfoque BBD, consistente en los pasos previos necesarios para crear un diseño previo a la sustentación; el lenguaje de representación de las abstracciones L-BBD, que permite expresar el producto a nivel de diseño.

7.2.1. Enfoque BBD

Como método de ingeniería de software orientado al abordaje de los niveles de abstracción y descomposición involucrados en el pensamiento computacional por parte de niños y adolescentes, Block-Based Design utiliza una estrategia de diseño top-down, desde los aspectos generales de los escenarios de interacción hasta lo más particular que son las acciones, englobando los sensores y demás disparadores a través de los que se dan éstas; el objetivo aquí es confirmar, a través de un estudio de caso, los niveles de abstracción que concuerden o correspondan con aquellos caracterizados y facilitados por el método en su transición del diseño a la implementación en un entorno visual de programación como Scratch.

A continuación, se describe los pasos que los niños siguen durante el Enfoque - BBD y los aspectos de representación tendientes a la definición de los constructos del diseño que pueden ser explotados más adelante como lenguaje de modelado. En la figura 9 se muestra la secuencia de pasos.

Figura 9. Pasos del enfoque BBD



Fuente. Propia de la investigación

Análisis inicial de la misión: leer detenidamente la misión en todo el equipo y discutirla durante 15 minutos.

En esta fase inicial del proceso de ChildProgramming, el primer paso es la explicación del objetivo de la misión de cada uno de los equipos de trabajo, comunicando a todos los roles, los tiempos para realizar la práctica, el entorno de programación, los elementos que se tienen a disposición.

Luego, se resuelven las inquietudes de cada uno de los grupos realizando rondas que permiten atender a cada uno de sus integrantes y a todo miembro de la comunidad de aprendizaje, siendo pertinente el énfasis en la forma en que se instala Scratch, la manera en que se puede acceder a cada uno de sus componentes u objetos.

A continuación, se explican los diferentes objetos que se pueden emplear en los niveles de abstracción inicial como mensajes, personajes, escenarios, disfraces o eventos y se sugiere un trabajo en equipo para que se pueda cumplir a cabalidad con la misión, partiendo desde una comprensión cabal de los diferentes niveles de comprensión del requerimiento planteado.

Identificar y clasificar objetos: leer detenidamente la misión identificando el ambiente de la misión (potenciales escenarios), sustantivos relevantes (potenciales objetos) y las acciones.

El primer paso en esta fase es la identificación de los objetos que están implicados en cada uno de los procesos que se requieren para el cumplimiento de la misión, y la clasificación en función de su naturaleza.

Entonces, se busca caracterizar cada uno de estos objetos en cuanto a sus propiedades por su color, forma o su posición dentro del escenario, así como su función específica dentro de la representación visual que sirve de contexto para las acciones necesarias que involucran estos objetos.

Finalmente, se hace una representación simbólica de estos objetos, de sus propiedades, y su posición dentro del del contexto visual del programa, lo que implica desplegar aspectos de transformación perceptual.

Organizar en función de las acciones y relaciones con otros objetos: esto busca definir las interacciones entre los objetos y las respuestas a estas interacciones.

Una vez se tiene la identificación y clasificación de los objetos requeridos se puede pasar establecer las diferentes clases de objetos, y a organizarlos en función de las acciones que desempeñan con relación a otros objetos lo que implica analizar cuáles son necesarios para desplegar una

determinada acción.
Luego, cada uno de los objetos, por su clase, se puede clasificar de acuerdo a su función dentro de una acción particular, teniendo en cuenta a la meta u objetivo de esa acción cuál será su rol dentro del escenario.
Así, el paso final es tener una clara idea de cómo se despliega en cada una de las acciones de cuáles son los objetos involucrados en su desempeño, a fin que se pueda cumplir la meta propuesta y contribuir así a tener un entorno que responda las acciones que plantea el usuario dentro del juego.

Definir los inicios de acción: se estructuran las acciones y movimiento como respuesta a eventos y mensajes.
En esta parte se estructuran las acciones que se pueden tener y la forma en cómo se inician puede ser mediante mensaje, clic de ratón o una palabra o mediante la pulsación de una de las teclas de un teclado.
De igual manera se deben de que procesos se desencadenan al activar la acción como si se va tocar una pista determinada en audio, se va mostrar mensajes, si se va a salir del juego porque se violó una de las reglas, si se reinicia el juego etc.
También se cambia de forma dinámica en el color de fondo o los escenarios que se están mostrando en entorno del juego, si es posible hacer acciones multijugador o de un solo participante y como es la acumulación del puntaje cuando se ejecuta una determinada acción.

Escenarios, formas y colores: se discute y diseñan los escenarios considerando imágenes de fondo formas geométricas básicas y colores de acuerdo a la narrativa de la misión.

Objetos, personajes y rivales: se diseña gráficamente los objetos, personajes y rivales y se le asocian los comportamientos definidos previamente (acciones, eventos, mensajes).
En esta sección se despliegan las habilidades de los estudiantes para la representación gráfica de los personajes, los objetos, la narrativa visual.
Es necesario que se enfatice a los estudiantes sitios más básicos del diseño gráfico y que consiste en la simplicidad, reflejando para cada objeto y personaje lo más importante, al igual que en las opciones de movimiento que pueden realizar o si por el contrario se trata de entidades fijas.
De igual forma, redefinir la relación de interacción entre ellos es decir si son amigos o si son rivales

y cuál es su dominio del escenario por el contrario se van a poder mover toda la pantalla o el escenario activo del juego.

Operaciones lógico-matemáticas, tiempos y puntos: las reglas lógicas, temporales y de variables de conteo son definidas y asociadas al comportamiento de los personajes, escenarios y objetos.
El razonamiento lógico es parte fundamental para poder desplegar los ciclos y acciones pertinentes que permitan crear un entorno de juego interactivo que tenga la suficiente complejidad para mantener al jugador motivado.
De igual forma, se debe promover entre los estudiantes pautas de razonamiento lógico como son las pautas dadas por Polyá para la solución de problemas en matemáticas y que, son de gran valor cuando se aplican a la comprensión de un requerimiento computacional, puesto que constan en fases muy específicas: comprensión del problema; realización de una planeación, que implica buscar un patrón, aplicar ensayo y error, resolver una ecuación, aplicar razonamientos indirectos; al llevar a cabo el plan y hacer su verificación.
De igual manera, es importante establecer la manera en que se ha de hacer la temporización de cada uno de los eventos que están implicados en el programa, lo que implica el manejo de relojes, de ciclos específicos.

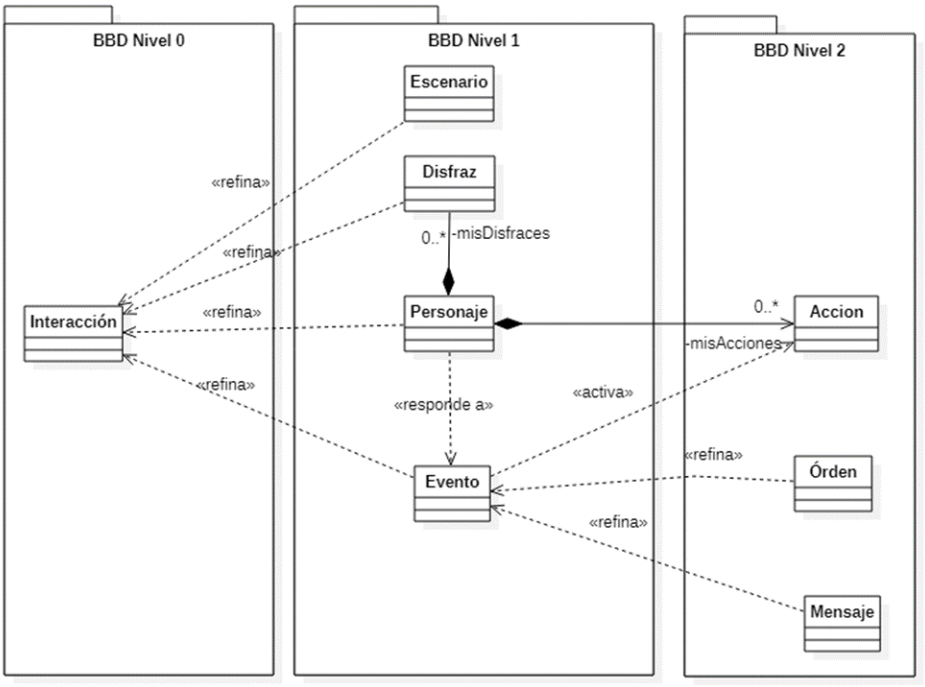
Uso de sensores: se definen los puntos desde donde se disparan las interacciones y se da inicio al cierto comportamiento global de la misión desde el mouse, el teclado o cualquier otro periférico.
En este contexto se tiene que mirar cuáles son los periféricos que exige el programa para su ejecución o para iniciar una acción determinada como puede ser si la entrada de datos va a ser por teclado o si por el contrario, se tiene la opción de activar objetos mediante clips de ratón.
De igual manera, si se necesita tener una cámara activa o si se necesita periféricos de su sonido, dado que en el juego se van a desplegar efectos musicales o midi.
También si se requiere algún otro tipo de sensor como por ejemplo o de movimiento o un ratón especial o si se requiere algún tipo de captura de información con un dispositivo móvil.

Siguiendo los pasos anteriores es posible pasar de la planeación a la implementación en el lenguaje, que es el entorno Scratch y realizar la implementación, superadas las dificultades referentes a operaciones del manejo de objetos como la clonación y el refinamiento top-down.

7.2.2. Elementos del Lenguaje de diseño del método BBD

El lenguaje de diseño busca brindar un espacio para delimitar los constructos de diseño sobre los cuales los niños describen sus soluciones en un alto nivel de abstracción. BDD-L cuenta con tres niveles de abstracción donde cada uno, a excepción del primero, refina los conceptos del anterior nivel en conceptos más específicos.

Figura 10. Elementos del Lenguaje BBD



Fuente. Propia de la investigación

7.2.3.1 Nivel de Abstracción 0

En el nivel inicial la misión es descompuesta en un conjunto de interacciones. Las interacciones son escenas en las que la misión logra un objetivo. Por ejemplo, hacer transitar un personaje por un laberinto. A continuación, se describen cada uno de los conceptos involucrados en este nivel:

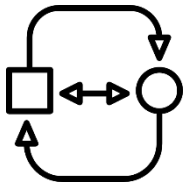
Concepto: Interacción

Descripción: describe una secuencia de acciones en una narrativa global (un escenario y varios

personajes) en la que se puede lograr alguna meta definida, por ejemplo, alcanzar un nivel, un premio, ganar o perder un juego, lograr un propósito.

Relaciones: ninguna

Representación (propuesta):



7.2.3.2 Nivel de Abstracción 1

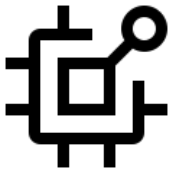
En este nivel las interacciones son refinadas con el fin de identificar los elementos del diseño clave y sus relaciones. El refinamiento de la interacción permite identificar escenarios, personajes, disfraces y los eventos de comunicación entre usuarios y personajes. Los constructos se describen a continuación:

Concepto: Escenario

Descripción: describe el ambiente sobre el cual se desarrolla la interacción, puede incluir objetos y sufrir transformaciones durante la interacción. Es posible que dada una interacción las variantes de un escenario se manejen como otro escenario.

Relaciones: Un escenario se refina en el siguiente nivel de abstracción en personajes, disfraces y eventos.

Representación (propuesta):



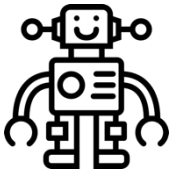
Concepto: Personaje

Descripción: describe un objeto que tiene "vida" en la misión, es decir tiene un comportamiento propio que puede ser autónomo o dado por los eventos del usuario. Ese comportamiento es una combinación de aspectos lógicos, algorítmicos y de interacción con otros personajes u objetos. Un personaje tiene asociado un disfraz (forma visual de representarse). El cambio de comportamiento

puede requerir del cambio de disfraces.

Relaciones: Un personaje puede tener varios Disfraces, Un personaje responde a los eventos asociados y se refina en el siguiente nivel con acciones que responden a esos eventos asociados.

Representación (propuesta):



Concepto: Disfraz

Descripción: describe una forma de visualizar un personaje

Relaciones: Un disfraz es parte de un personaje, Un personaje contiene con conjunto de acciones que describen el comportamiento frente a algún Evento al cuál se responde.

Representación (propuesta):

Concepto: Evento

Descripción: describe un activador de la interacción o un cambio en la interacción (puede ser una condición lógica, un mensaje o una orden desde un sensor)

Relaciones: Un evento en el siguiente nivel de abstracción se refina en mensajes y órdenes (desde los sensores). El evento activa una o varias acciones.

Representación (propuesta):

7.2.3.3 Nivel de Abstracción 2

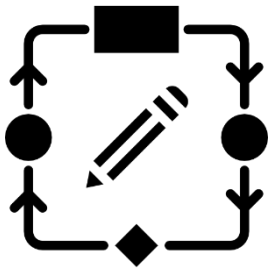
En este nivel el comportamiento de los personajes es refinado en acciones y los eventos refinados en mensajes y órdenes. A continuación, se describe cada uno de los constructos.

Concepto: Acción

Descripción: describe una actividad a realizar por parte del personaje como respuesta a un evento, normalmente se asocia a un verbo y se refina a nivel de programación como un algoritmo que permite automatizar la acción. A nivel de diseño, opcionalmente se puede esquematizar en forma general el algoritmo.

Relaciones: Refina un evento y por tanto activa alguna acción en el personaje receptor.

Representación (propuesta):



Concepto: Mensaje

Descripción: describe un activador de la interacción o un cambio en la interacción debido a un mensaje de un personaje a otro. Desde la acción de un personaje se envía un mensaje a otro quien tiene asociado una o varias acciones como respuesta.

Relaciones: Refina un evento y por tanto activa alguna acción en el personaje receptor.

Representación (propuesta):

Concepto: Orden

Descripción: describe un activador de la interacción o un cambio en la interacción debido a una orden desde un sensor (teclado, ratón u otro periférico).

Relaciones: Refina un evento y por tanto activa alguna acción de los personajes.

Representación (propuesta):

7.2.3. Sintaxis Concreta del Lenguaje

La sintaxis concreta (símbolos y espacio de diseño) se ha dejado abierta para que los niños utilicen

sus propias representaciones, el grupo IDIS espera en un siguiente proyecto de investigación indagar más sobre la representación y determinar si una sintaxis concreta podría resultar útil. Sin embargo, en el último apartado de las descripciones anteriores se hace una propuesta de representación que puede servir de punto de partida.

8. Evaluación del Método BBD a través de un Estudio de Caso

En este capítulo se expone la metodología que se utilizó para el desarrollo de este trabajo de grado investigativo realizado con niños de 8 a12 años. Incluyendo el contexto de la investigación y la especificación de los resultados.

8.1. Diseño del Estudio de Caso

Este estudio tiene el objetivo de evaluar la utilidad del método para facilitar aspectos claves en la metodología ChildProgramming a nivel de diseño: la abstracción y la colaboración. Así mismo evaluar la correspondencia de las soluciones con los constructos. Para ello se organizó un curso de programación con Scratch a través del CECAV de la Universidad del Cauca en el que se trabajó con niños entre 8 y 12 años de edad, organizados en equipos de 4 o 5 estudiantes con un total de 10 estudiantes. El estudio de caso es embebido, exploratorio y revelatorio.

Pregunta de investigación: Para conocer si existe una correspondencia entre la representación de la solución de problemas computacionales y los constructos caracterizados, este estudio de caso pretende resolver la siguiente pregunta de investigación

¿Existe una correspondencia entre la representación de las soluciones computacionales y los modelos mentales en los niños de 8 a 12 años del curso de programación del CECAV de la Universidad del Cauca?

Objetivo del Estudio de Caso: Confirmar la correspondencia entre el método BBD y las representaciones computacionales a la solución de problemas planteados en un paradigma de programación basado en bloques.

Instrumentos de Evaluación:

Dentro de este estudio de caso los instrumentos empleados aportan datos que posteriormente son analizados y evaluados para dar validez a la información recogida.

Para esta investigación, los instrumentos seleccionados son:

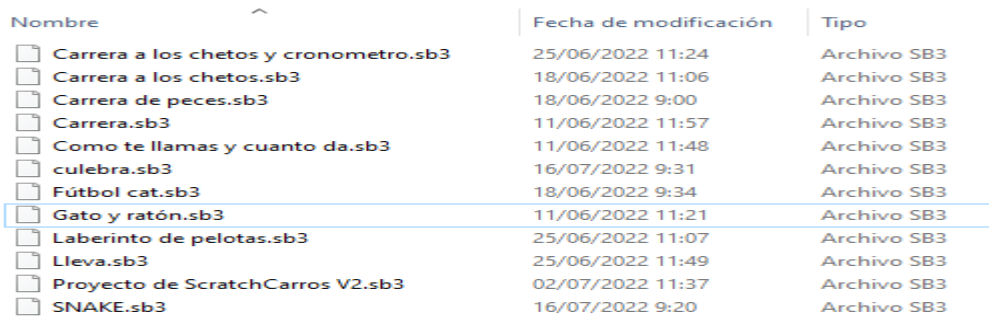
- **Observación de campo:** este instrumento establece una relación concreta e intensiva entre el equipo de investigación y el hecho social o los actores sociales de los que se obtienen datos que luego se sintetizan para desarrollar la investigación. La observación es un procedimiento de recolección de datos e información que consiste en utilizar los sentidos para observar hechos y realidades sociales presentes y a la gente donde desarrolla normalmente sus actividades. La observación de campo se registra en el Anexo B.

- **Planilla de registro** de representaciones, abstracciones y concreciones de código y su correspondencia.
- **El código Scratch**: Código resultante de las practicas.

8.2. Prácticas realizadas

Se realizaron 6 sesiones sabatinas, con una intensidad horaria de 4 horas, comenzando el 11 de junio y terminando el 16 de julio de 2022, (11,18, 25 de junio;2, 9 y 16 de julio), inicialmente con 14 estudiantes, de diferentes colegios de la ciudad de Popayán, con edades entre 8-12 años; el proceso fue finalizado por 10 estudiantes. Las prácticas se llevaron a cabo en el salón 333, sala de sistemas de la facultad de Ingeniería Electrónica de la Universidad del Cauca, Popayán. En ellas se llevaron a cabo las practicas que se observan en la figura 11.

Figura 11. Practicas realizadas en Scratch en el grupo del estudio de caso

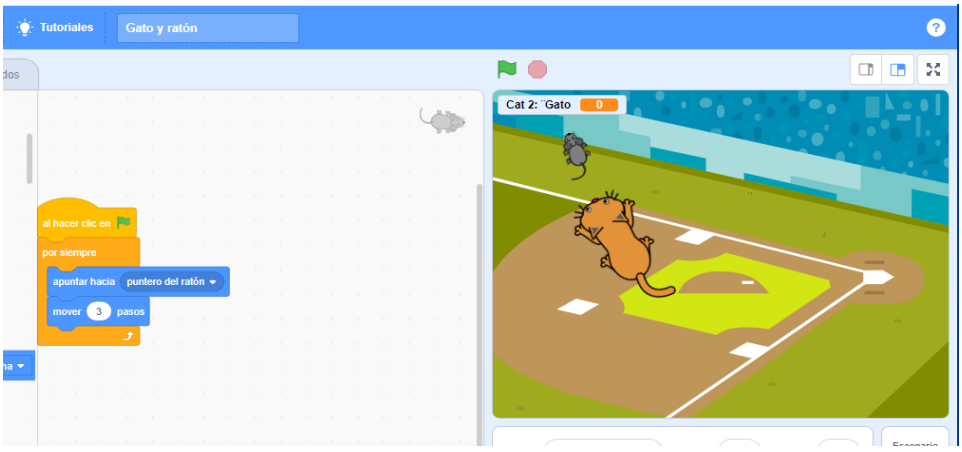


Nombre	Fecha de modificación	Tipo
Carrera a los chetos y cronometro.sb3	25/06/2022 11:24	Archivo SB3
Carrera a los chetos.sb3	18/06/2022 11:06	Archivo SB3
Carrera de peces.sb3	18/06/2022 9:00	Archivo SB3
Carrera.sb3	11/06/2022 11:57	Archivo SB3
Como te llamas y cuanto da.sb3	11/06/2022 11:48	Archivo SB3
culebra.sb3	16/07/2022 9:31	Archivo SB3
Fútbol cat.sb3	18/06/2022 9:34	Archivo SB3
Gato y ratón.sb3	11/06/2022 11:21	Archivo SB3
Laberinto de pelotas.sb3	25/06/2022 11:07	Archivo SB3
Lleva.sb3	25/06/2022 11:49	Archivo SB3
Proyecto de ScratchCarros V2.sb3	02/07/2022 11:37	Archivo SB3
SNAKE.sb3	16/07/2022 9:20	Archivo SB3

Fuente. Propia de la investigación

La primera practica es una animación de gato y ratón, donde se maneja sonido y condicionales simples, de manera que el gato persigue al ratón mientras no se pare la ejecución. Ver figura 12

Figura 12. Gato y ratón



Fuente. Propia de la investigación

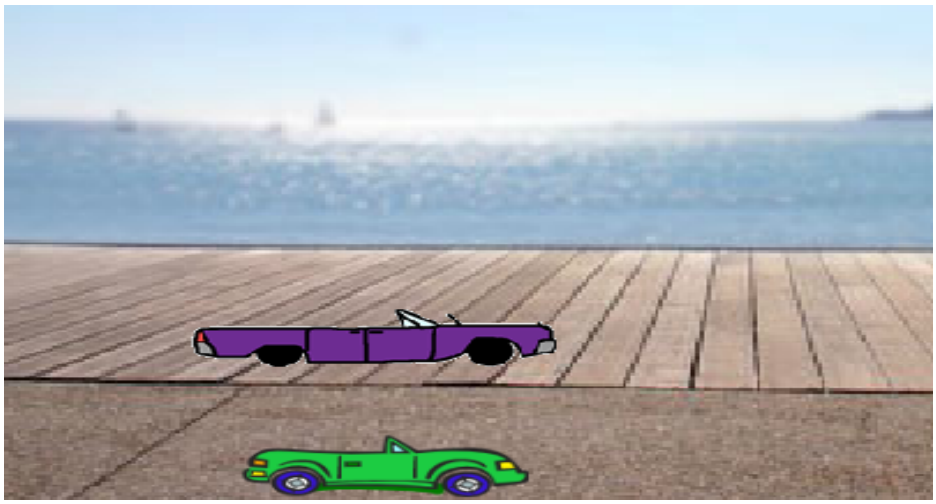
La segunda práctica consiste en el despliegue de mensajes (hola, como te llamas, dame un número, dame otro número da, adiós), utilizando el ratón y el teclado, de manera que se suman dos números dados y se da el resultado. Ver figura 13.

Figura 13. Suma de números de forma interactiva

Fuente. Propia de la investigación

La tercera practica remite a una carrera de dos autos, utilizando para la animación del movimiento números aleatorios entre 1 y 10. Ver figura 14.

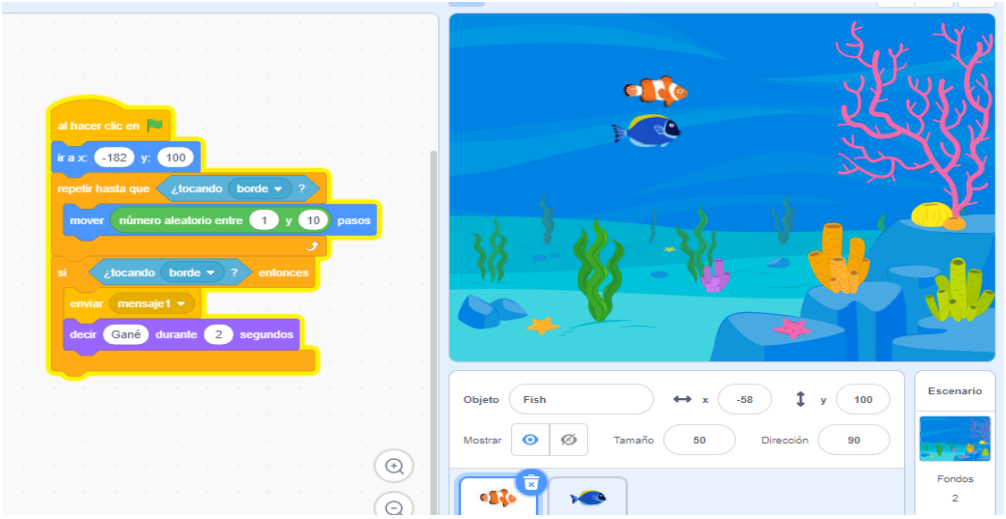
Figura 14. Carrera de autos



Fuente. Propia de la investigación

La cuarta practica refiere a una carrera de peces, programa que explora no solo sonidos, despliegue de mensajes como perdí o gané, sino también acciones de movimiento y ciclos repetir hasta que, propios de la programación. Ver figura 15.

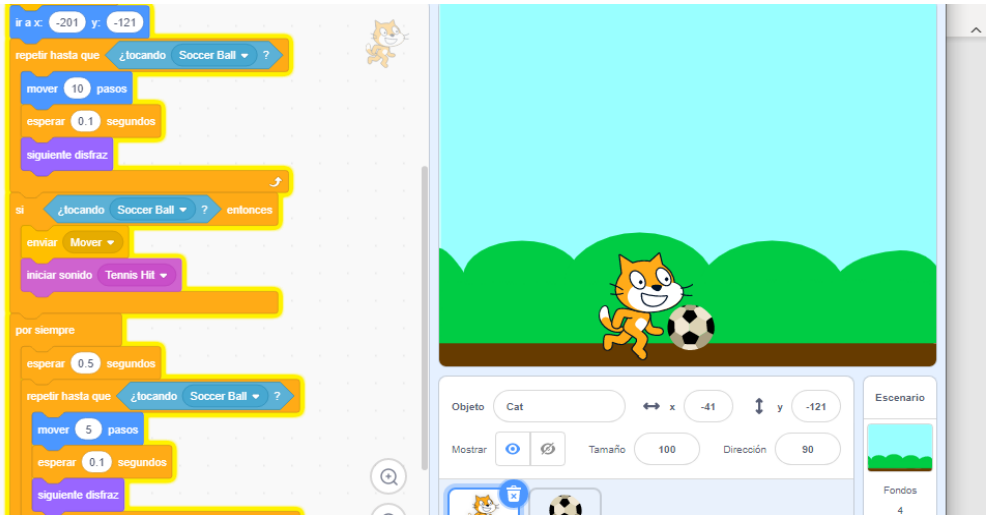
Figura 15. Carrera de peces



Fuente. Propia de la investigación

En la quinta práctica, de igual forma, se maneja sonidos, movimiento y el ciclo repetir hasta que, gestando una animación de un gato pateando un balón que rebota al llegar al extremo y que es vuelto a golpear hasta quedar quieto. Ver figura 16.

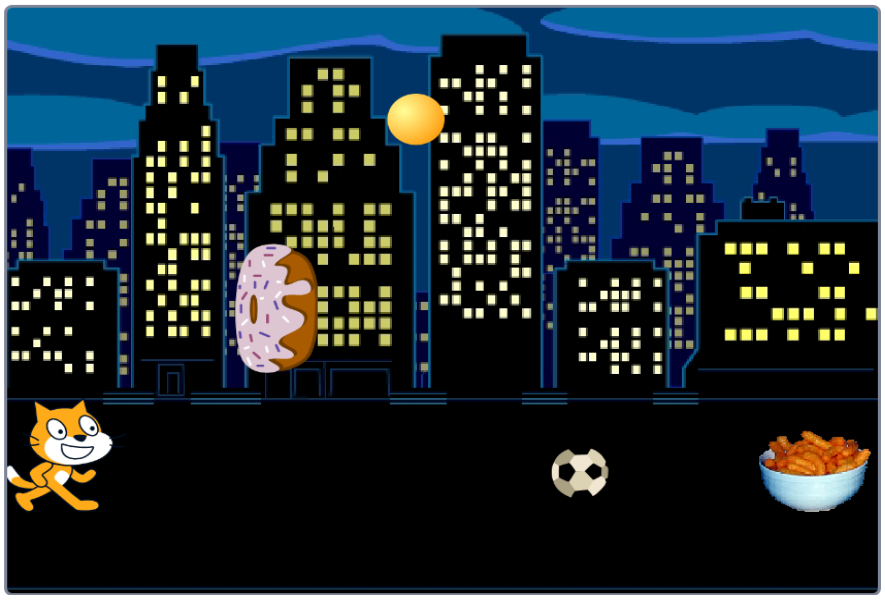
Figura 16. Gato y balón



Fuente. Propia de la investigación

En la sexta práctica llevada a cabo, se tiene una carrera a los chetos, articulado en ciclos condicionales anidados, lo que permite manejar el movimiento de múltiples objetos. Ver figura 17.

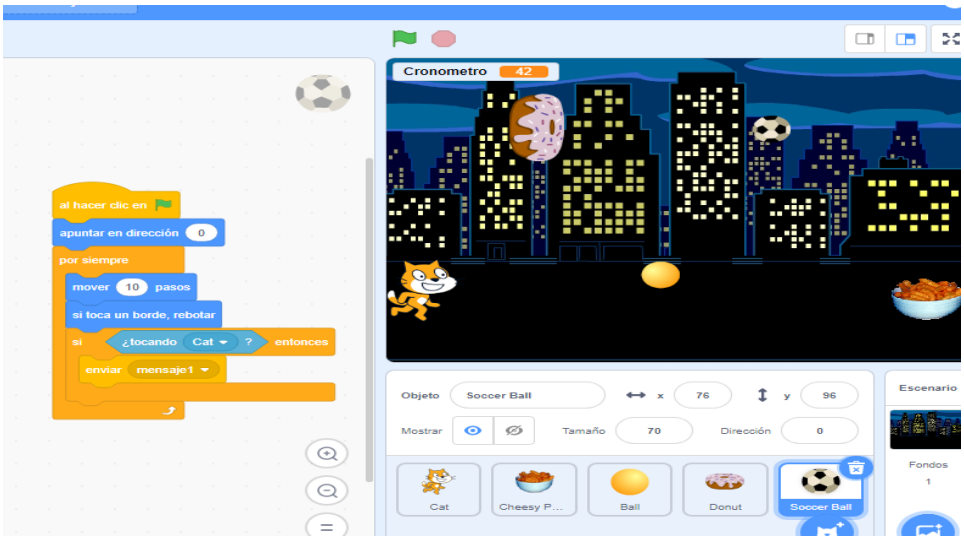
Figura 17. Carrera a los chetos



Fuente. Propia de la investigación

En la séptima actividad se añade a la carrera a los chetos un cronometro, introduciendo contadores en la programación. Ver figura 18.

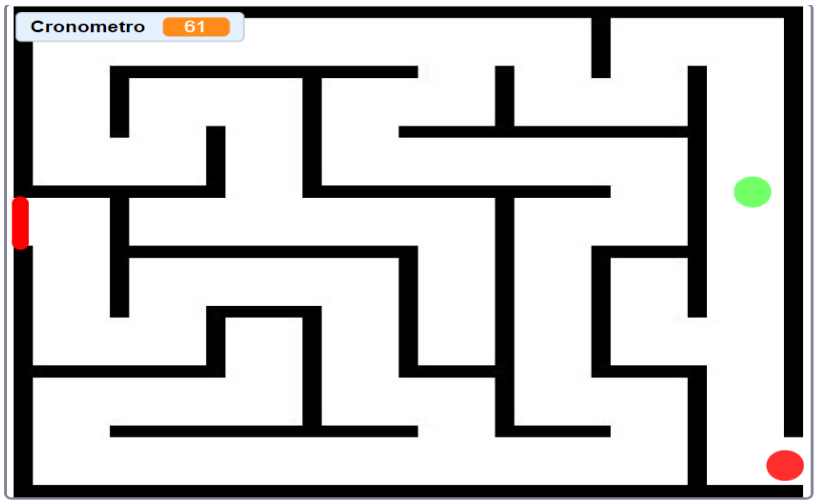
Figura 18.Cronometro



Fuente. Propia de la investigación

La octava actividad consiste en un laberinto, juego que tiene por objeto llevar un punto desde la entrada a la salida, con empleo de las teclas de dirección del teclado, en el menor tiempo posible; se tiene incorporado el cronometro y si chocas con alguna pared, retornas a la salida, esta actividad se jugó por parejas con dos integrantes paralelamente, el circulo verde y el círculo rojo. Ver figura 19.

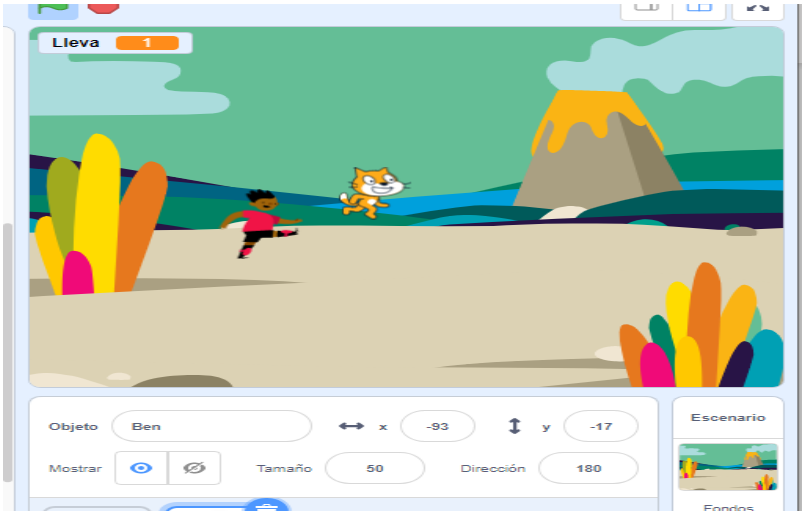
Figura 19. Laberinto.



Fuente. Propia de la investigación

En la novena actividad, denominada lleva, se tiene un gato que persigue un objeto en forma de humano, y el cronometro indica el tiempo empleado en la captura; el humanoide se mueve con las teclas arriba, abajo, derecha e izquierda del teclado. Ver figura 20.

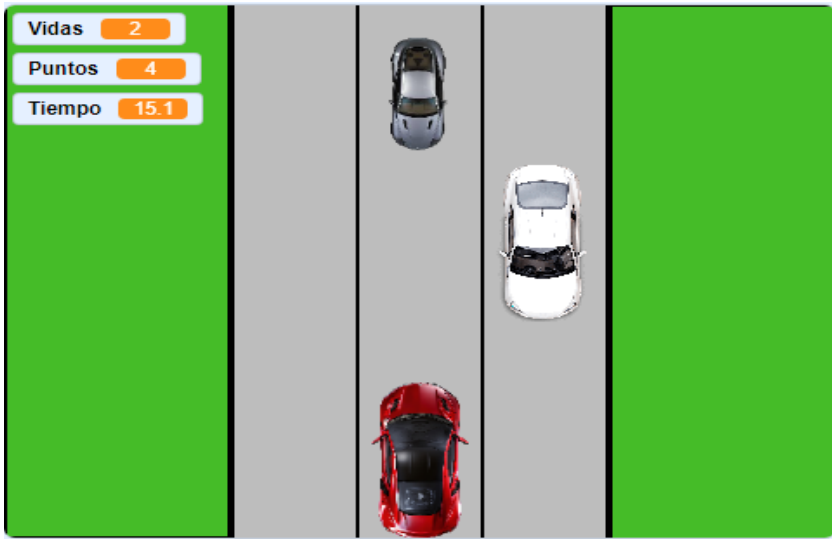
Figura 20. Lleva.



Fuente. Propia de la investigación

La décima actividad es un juego más elaborado consistente en una carrera de carros, evitando ser chocado por los rivales abarcando ciclos, contadores, funciones y acumuladores. Se indica el tiempo, puntaje y vidas que se tienen. Ver figura 21.

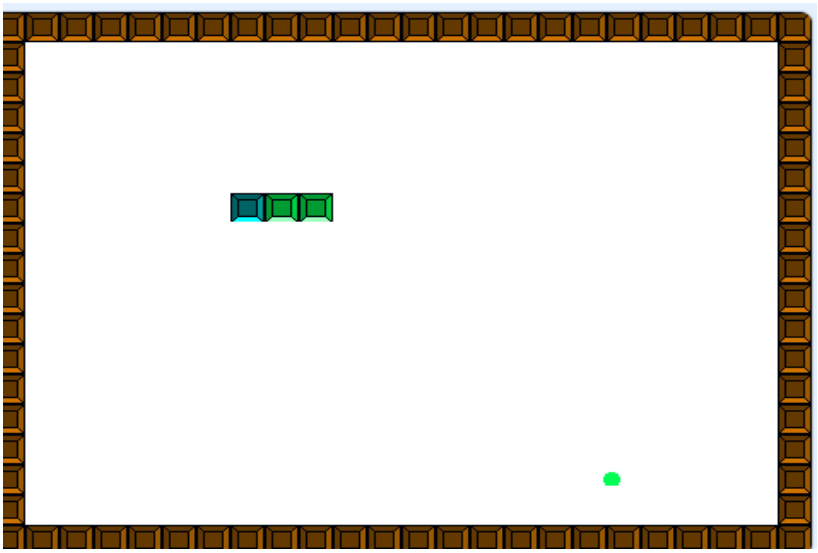
Figura 21. Coches



Fuente. Propia de la investigación

La undécima actividad remite a una serpiente a la que debe alterarse su movimiento mediante las teclas de dirección del teclado. Ver figura 22.

Figura 22. Serpiente



Fuente. Propia de la investigación

8.3. Análisis de las métricas utilizadas

Los estudiantes involucrados en el estudio de caso trabajaron en parejas, de modo que se formaron 5 grupos; ninguno de los estudiantes tenía experiencia previa con la programación, ya que provenían de diferentes instituciones educativas públicas, que, dentro de su área de informática y tecnología, con su intensidad horaria de 2 horas semanales, ilustran las relaciones de tecnología y sociedad y no tanto la programación en sí. Ver figura 23.

Figura 23. Participantes del estudio de Caso

Fuente. Propia de la investigación

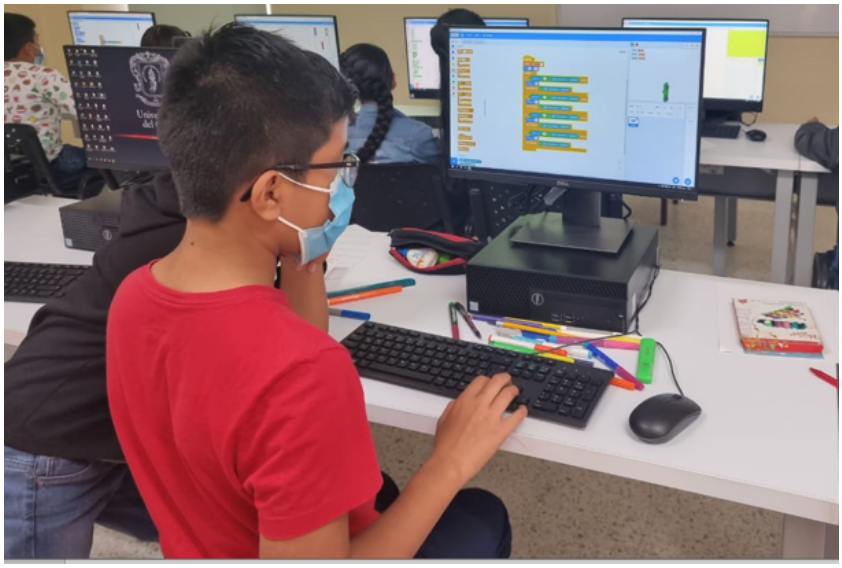
En concordancia con la metodología elegida de ChildProgramming, es posible evaluar si la utilización de Scratch, promueve el pensamiento computacional. Para realizar esta tarea es necesario identificar pautas de la forma en que analizan y formulan el problema los estudiantes, lo que remite al trabajo en equipo. Ver figura 24.

Figura 24. Análisis, formulación y trabajo en equipo

Fuente. Propia de la investigación

Así mismo, como parte del proceso de abstracción, incrustado dentro del pensamiento computacional, se tiene el manejo de estructuras que reflejan un pensamiento reflexivo como son las funciones y ciclos, que se han utilizado dentro de las prácticas realizadas. Ver figura 25.

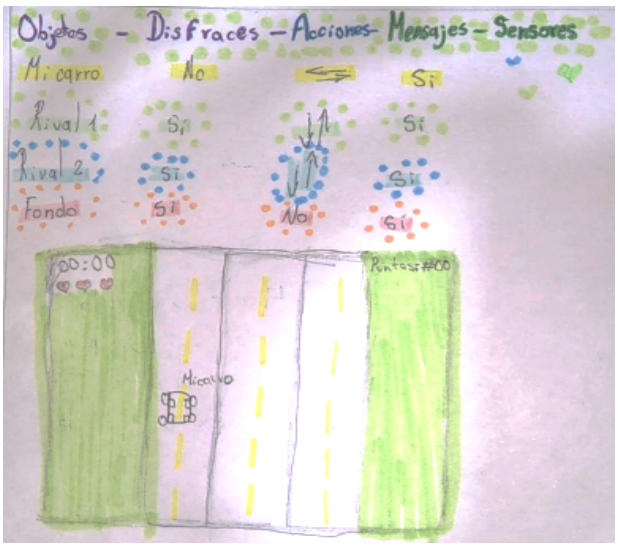
Figura 25. Abstracción reflexiva



Fuente. Propia de la investigación

Así mismo, clasifican lo que es útil para su algoritmo de lo que no es importante, y representan gráficamente la situación, utilizando objetos y clases. Ver figura 26.

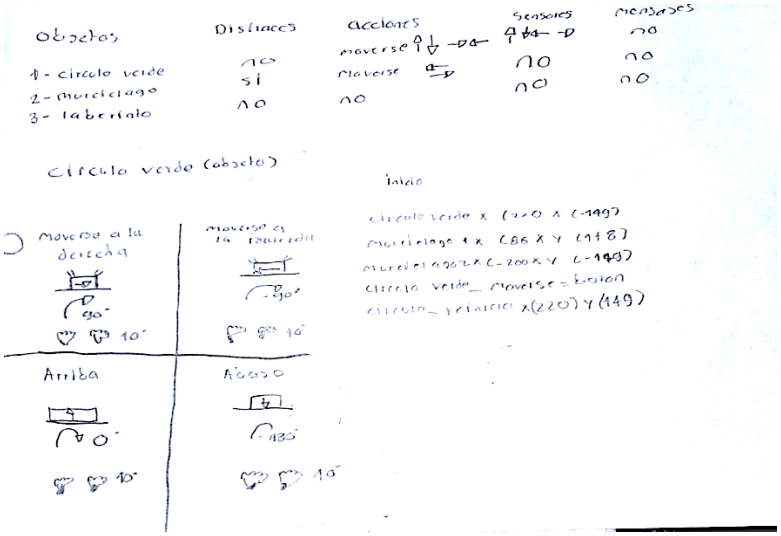
Figura 26. Abstracción, objetos y clases



Fuente. Propia de la investigación

De igual manera, el cambio de representación, la transformación perceptual, se muestra en los esquemas conceptuales e imágenes sobre el problema a resolver, los refinamientos, utilizados por los estudiantes. Ver figura 27.

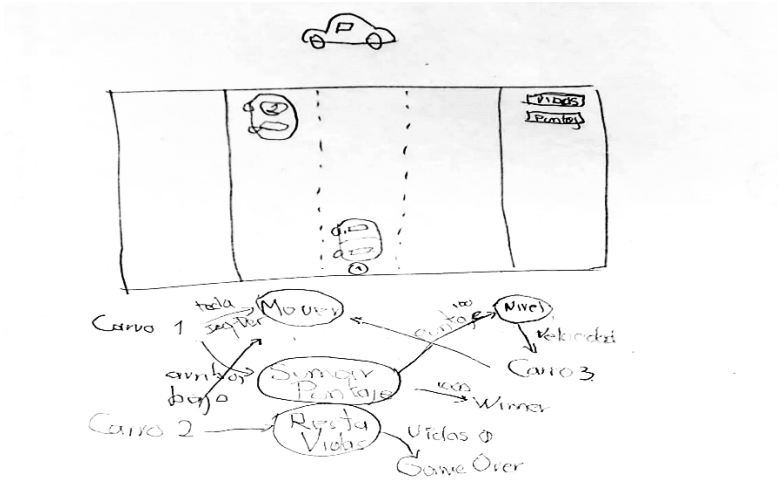
Figura 27. Transformación perceptual



Fuente. Propia de la investigación

Igualmente, se presenta la abstracción como mapeo sintáctico y simbólico, debido a que en el pensamiento computacional deviene un método en el que se representa la información mediante objetos y estructuras visuales. Ver figura 28.

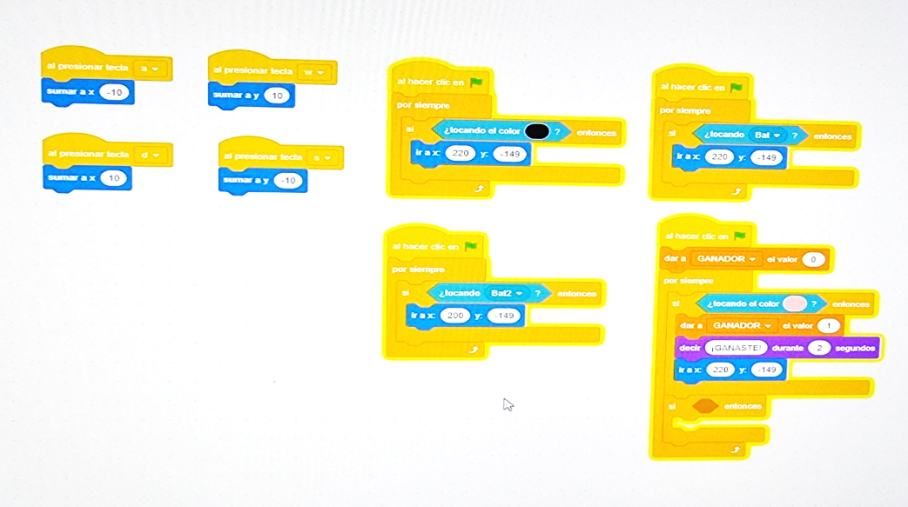
Figura 28. Abstracción como mapeo sintáctico y simbólico



Fuente. Propia de la investigación

Se percibe también la abstracción como herramienta de diseño conceptual de modelos porque permite reducir el número de objetos, pero también construir módulos, es decir, descomponer la complejidad. Esto se puede ver en la manera que los estudiantes desarrollan los bloques. Ver figura 29.

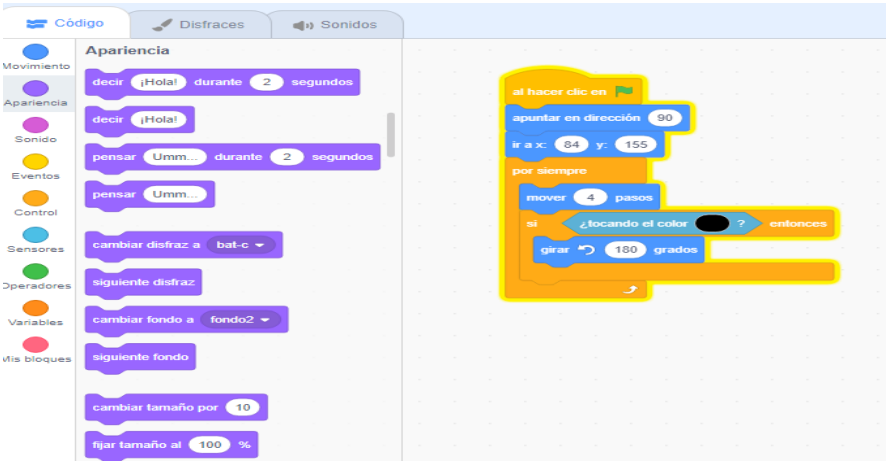
Figura 29. Diseño conceptual



Fuente. Propia de la investigación

También, existe la posibilidad de depurar lógicamente el código, lo que facilitar el aprendizaje de los estudiantes. Ver figura 30.

Figura 30. Depuración lógica del código

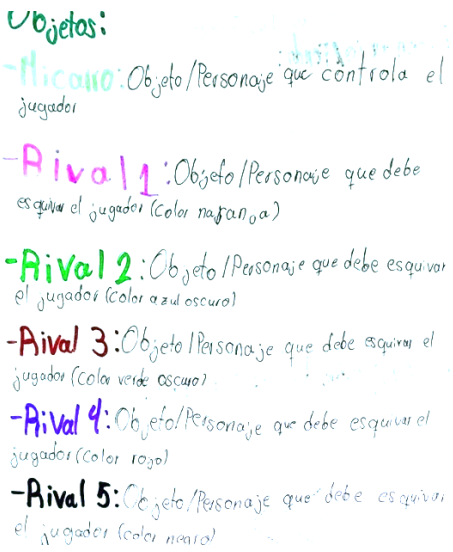


Fuente. Propia de la investigación

Dado que la abstracción implica analizar, descomponer, diseñar, estructurar, evaluar y depurar algoritmos que realicen tareas específicas, requiere que esta experiencia , consistente en desplegar paso a paso las instrucciones, el código, el pensamiento computacional, sea divertido, motivador

y eso se logra a través de la lúdica , que simplifica el análisis y diseño, mediante objetos específicos, cuyos atributos son la base de la abstracción sintáctica que despliegan los estudiantes del grupo del caso exploratorio. Ver figura 31.

Figura 31. Análisis sintáctico de objetos

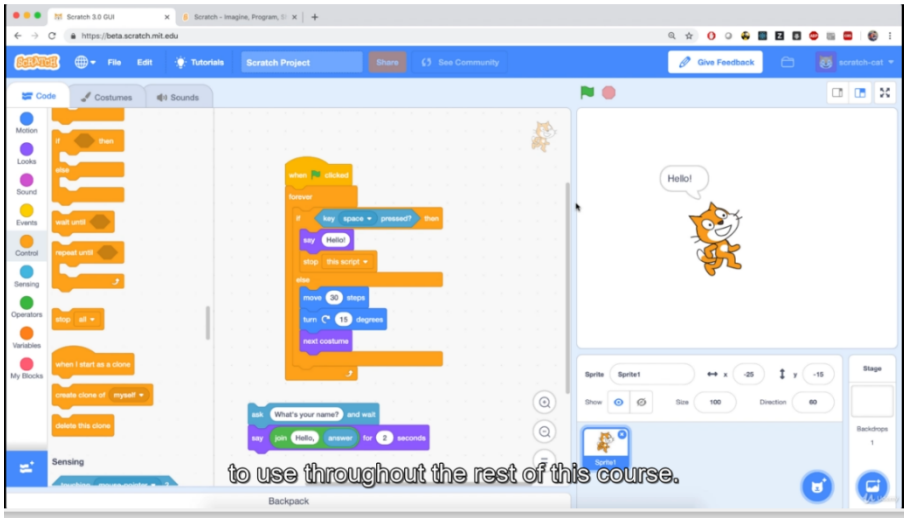


Fuente. Propia de la investigación

La gamificación, como estrategia, es fundamental con los programas visuales y, por supuesto, Scratch está diseñado para ella; así, posee bloques para movimiento, sonido y escenarios; involucra herramientas de diseño, iteraciones, procedimientos, despliegue de mensajes, contadores, cronómetros o relojes análogos, lo que constituye una introducción a la programación visual con interacción de los estudiantes, la ciencia, el arte, las matemáticas y por supuesto los juegos (Marji, 2014).

En este orden de ideas, a esta labor de gamificación y de despliegue de la programación visual basada en bloques, con privilegio de la abstracción, en sus diferentes niveles, y textos sobre la temática como los de (McManus, 2009),(Bourret, 2015), (Woodcock, 2016) y (Vlieg, 2016), son un gran soporte; de igual manera existen cursos en línea de gran impacto como los de Udemy; en la figura 32 se muestra el entorno de uno de estos cursos.

Figura 32. Cursos en línea



Fuente. https://www.udemy.com/course/scratch-programming/

8.4. Discusión de las métricas

En cuanto a la evaluación de las métricas, a partir del rendimiento de los estudiantes del estudio de caso, se generan la estadística descriptiva que aparece en la tabla 2.

Tabla 2. Estadística de las métricas en el caso exploratorio

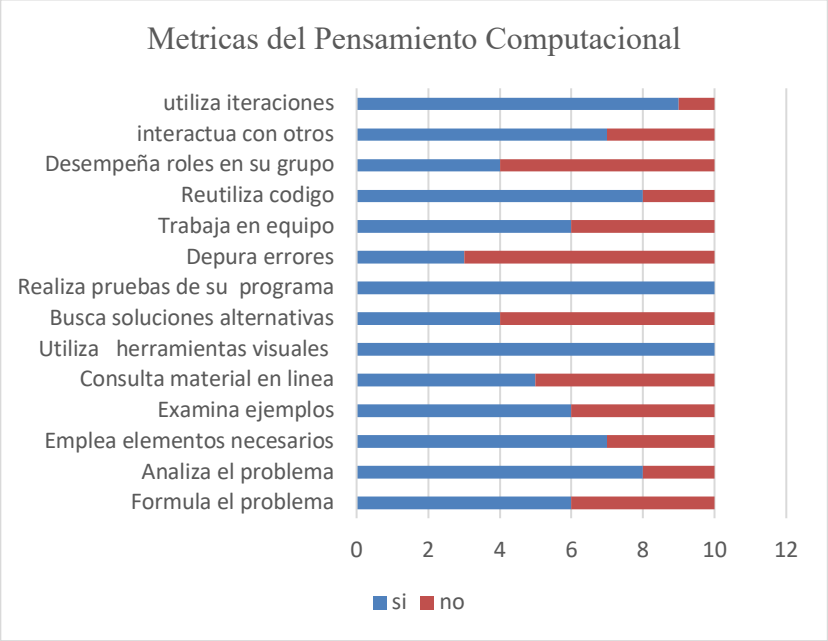
Codificación	Ítem	Opción	
		Si	No
1	Formula el problema	6	4
2	Analiza el problema	8	2
3	Emplea elementos necesarios	7	3
4	Examina ejemplos	6	4
5	Consulta material en línea	5	5
6	Utiliza herramientas visuales	10	0
7	Busca soluciones alternativas	4	6
8	Realiza pruebas de su programa	10	0
9	Depura errores	3	7
10	Trabaja en equipo	6	4
11	Reutiliza código	8	2
12	Desempeña roles en su grupo	4	6
13	Interactúa con otros	7	3
14	Utiliza iteraciones	9	1

Fuente. Propia de la investigación

En la figura 33 se grafican las métricas de la experiencia realizada, lo que permite evaluar las destrezas de pensamiento computacional alcanzadas con la aplicación del ChildProgramming a

través de Scratch.

Figura 33. Métricas de pensamiento computacional en el caso exploratorio



Fuente. Propia de la investigación

Entonces, se puede afirmar que: el 60% de los participantes pueden formular el problema;80% analiza la situación planteada y 70% es capaz de abstraer los elementos necesarios para realizar la programación visual. Obviamente, 100% utiliza herramientas visuales, aunque el empleo de ayudas en línea o de buscar soluciones alternativas o ejemplos, solo se presenta en 50%,60% y 40% de los casos. Las iteraciones se utilizan en 90% de los casos

La depuración de errores lógicos es hecha solo por el 30% de los estudiantes, si bien todos prueban o ponen a funcionar sus programas. 60% trabajan eficientemente en grupo y 40% son eficientes en desempeñar un rol de liderazgo; 70% interactúa con los otros grupos y 80%, en consecuencia, reutiliza códigos.

Se debe anotar que los ítems correspondientes a la categoría de abstracción se colocan de color gris y los de colaboración, en rojo, en la tabla 2. Con base en ello, los resultados positivos y negativos por categoría, del total posible de 70 puntos, tanto para sí como para no, se pueden sumar y con ello elaborar una tabla de doble entrada para estas categorías. Ver la tabla 3.

Tabla 3. Categorías del pensamiento computacional

Categoría	Si	No	Total

Abstracción	53	17	70
Colaboración	40	30	70
Total	93	47	140

Fuente. Propia de la investigación

Es posible afirmar entonces que, dentro del estudio de caso se confirma que a través de ChildProgramming se obtiene una mejora importante de las destrezas de abstracción de los niños, y una buena mejora de las capacidades de trabajo en equipo y colaboración.

En general, se confirma que la estrategia ChildProgramming añade a las pautas o niveles de abstracción la descomposición, la interacción y la generalización, además de utilizar la gamificación para aumentar la productividad y el pensamiento computacional, existiendo en general entre los grupos un nivel más elevado de abstracción que de colaboración, destacándose en equipo 2 como el de mejores resultados. Ver tabla 4.

Tabla 4. Pensamiento computacional en los equipos

Equipo	Abstracción	Colaboración
1	13	7
2	15	9
3	10	7
4	8	10
5	7	7
Total	**53**	**40**

Fuente. Propia de la investigación

Respecto a la correspondencia entre el lenguaje brindado y su uso durante los procesos de diseño en la siguiente tabla se resumen los principales hallazgos.

Tabla 5. Correspondencia de BBD con los modelos de diseño creados

Constructo	Nivel de Correspondencia	Justificación
Interacción-Escenario	Media	Presente visualmente en todos los casos como una imagen contenedora de todos los elementos involucrados. Sin embargo, no es explícitamente nombrada ni separada de otras interacciones que pueden ocurrir en el mismo escenario por que puede inferirse es la interacción primaria de la misión.
		Presente visualmente en todos los casos como una imagen contenedora de todos los elementos involucrados. Sin embargo, no es explícitamente nombrado en la misión.
Personajes y objetos	Alta	En todos los trabajos se identifican con la mayoría de relaciones expresadas en BBD
Disfraces	Baja	Presentes explícitamente en dos de los cuatro modelos descritos por los equipos de estudiantes

Acciones	Alta	Descritas en todos los modelos de forma explícita, normalmente a través de íconos que representan la acción misma como por ejemplo moverse a la derecha
Eventos, Mensajes y Ordenes	Media	Descritos en algunos de los trabajos, pero hay que considerar que no en todos se hizo uso de los mensajes como forma de interacción, están presentes las órdenes (asociadas a movimientos y acciones) a través de flechas, en cuadros

Fuente. Propia de la investigación

Revisando estos resultados se puede deducir que hay una adecuada correspondencia con el modelo propuesto y se sugieren algunas mejoras:

Proponer un lenguaje gráfico que facilite la representación

Proponer un espacio de diseño que facilite ir de lo general a lo específico

Poner énfasis en el diseño de eventos, mensajes y órdenes para diferenciar y poder razonar de manera abstracta sobre las dinámicas presentes en la misión

Algunas de estas mejoras fueron consideradas en el refinamiento del proceso y del lenguaje de BBD.

Las métricas encontradas evidencian que los estudiantes, a nivel cualitativo son capaces de interactuar en un nivel medio con el escenario y que lo conciben de una forma modesta como imagen contenedora de los elementos que componen la misión, de manera que no está claramente definido a nivel gráfico el escenario del juego de acciones concomitantes que pueden cambiar la apariencia del mismo al realizar una interacción específica.

De igual forma, existe un alto nivel de desempeño en cuanto a la competencia de representar mediante íconos las acciones que se realizan dentro del entorno del juego, como por ejemplo una flecha a la derecha para indicar movimiento en esa dirección, un parlante para indicar sonido, un reloj para indicar tiempo, etc.

El rendimiento en cuanto a la utilización de eventos mensajes y órdenes está en el nivel medio y remite a que no todos tienen la misma comprensión lectora de las exigencias implicadas en la misión; de igual manera es muy baja la elaboración de personajes con disfraces, lo que implica que la caracterización de determinados objetos en cuanto a las propiedades multimedia y las capacidades para elaborar storyboards son relativamente modestas.

9. Conclusiones

Se realizó una revisión de la dura existente en 105 documentos, de los cuales 84 son artículos científicos, 19 libros especializados y dos proyectos de tesis, utilizando como palabras clave pensamiento computacional, abstracción programación basada en bloques, diseños software, colaboración, Scratch, que son los conceptos desarrollados en el marco teórico; la búsqueda se realizó fundamentalmente en Google académico.

Se eligió como metodología de diseño software ChildProgramming puesto que el énfasis de este estudio es la abstracción facilitada mediante la programación visual basada en bloques, con herramientas libres como Scratch, como se percibió en diferentes fuentes de la revisión de la literatura que privilegia los marcos metodológicos en los que los niños y jóvenes puedan programar sus juegos, sus narrativas, con facilidad de uso, en tanto que aprendían conceptos como bucles, iteraciones, algoritmos y se adentraban en el pensamiento computacional, la abstracción, la recursión, la modularidad, destrezas de resolución de problemas, aproximándose a la puesta en práctica de la espiral del aprendizaje creativo de Resnick:*imaginar, crear, jugar, compartir, reflexionar, imaginar.*

Puesto que ChildProgramming, como metodología fundamentada en la programación visual basada en bloques, busca responder la manera en que los equipos de trabajo de niños de 8-12 años utilizan la abstracción, la forma en que colaboran y produce, y si son capaces de realizar sus propios juegos, lo que implica el dominio de los conceptos básicos de la programación estructurada, y la puesta en práctica del pensamiento computacional, se realizaron 6 sesiones de 4 horas con 10 niños de 8-12 años, como caso exploratorio, donde los equipos de 2 personas, realizaron 11 proyectos o prácticas que implicaron análisis, diseño, abstracción, manejo de ciclos, de sonido, de contadores, de recursión, despliegue de mensajes, entre otros.

De las prácticas del estudio de caso se determinó que los estudiantes desplegaron destrezas de abstracción: con objetos y clases, transformación perceptual, semántica y simbólica, diseño conceptual; igualmente algunas acciones de depuración lógica del código y análisis sintáctico de objetos.

En concordancia con los resultados del estudio de caso se puede afirmar que ChildProgramming favorece las destrezas de abstracción y de cooperación, si bien en la actual experiencia, los

resultados fueron más importantes para la primer categoría; con ello, se puede concluir que la programación basada en bloques y herramientas como Scratch, promueven, mediante metodologías como ChildProgramming diversos niveles de abstracción en niños de 8-12 años, al igual que aceptables desempeños colaborativos.

En general, se confirma que el método BBD añade a las pautas o niveles de abstracción la descomposición, la interacción y la generalización un nivel más elevado en el pensamiento computacional.

Existen grandes falencias en las instituciones educativas en el ámbito de la computación en especial en niños de primaria, Aunque presentan gran interés por las ciencias de la computación no cuentan con profesores capacitados ni las herramientas tecnológicas para motivar a los niños por la ingeniería de software.

9.1. Trabajos Futuros

En cuanto a los estudios futuros, se propone que, con el objetivo de mejorar las competencias de pensamiento computacional de los estudiantes, se haga énfasis en los mecanismos de transferencia desde la representación de los requerimientos planteados, mediante un lenguaje gráfico, hacia la implementación en el entorno visual de programación de en los eventos, mensajes, órdenes y estrategias, que desde los diferentes mecanismos de abstracción se han determinado para la solución de los requerimientos planteados, siguiendo la idea del top-down, que va desde lo general a lo particular.

Así, para esta tarea se pueden tomar las líneas de trabajo planteadas por García & Orejuela (2014) y Pelánek & Effenberger (2022), que plantean estrategias de ramificación apropiadas para el entorno visual de programación basada en bloques, enfatizando en el diseño y análisis de micro mundos y rompecabezas, que se pueden utilizar para enriquecer la propuesta hecha en este documento y avanzar hacia el desarrollo de competencias de abstracción, mucho más eficiencia por parte de los estudiantes que se involucran en los entornos de programación basada en bloques y que pueden aprovecharlas las bondades del entorno de desarrollo visual de Scratch.

Así las cosas, en el futuro de lo que es posible realizar es una comparación más elaborada de la solución propuesta en este documento, basada en Scratch, con énfasis en la abstracción y otras

alternativas de entorno de programación visual basada en bloques, comparando las métricas que tienen que ver con la capacidad de representación simbólica de los procesos involucrados en la solución de problemas planteados, la comprensión de textos, la manera en que se utilizan las herramientas visuales de Scratch, entre otras.

En particular, se propone que lazos competencias de abstracción desarrolladas por los estudiantes con base en el método planteado aquí, se pongan a prueba cuando ellos trabajan en otros entornos de programación visual o en ambientes híbridos, con la finalidad de establecer las bondades del enfoque propuesto o las dificultades que tengan los estudiantes ante el cambio de entorno de programación, siendo esto un indicador de la efectividad de la adopción de la filosofía de la programación ágil por parte de los estudiantes.

De igual manera, se pueden incluir en la metodología de programación basada en bloques con el objetivo de mejorar las competencias de pensamiento computacional relacionadas con la abstracción, las ideas planteadas por Savidis (2022) que consideran los entornos de programación visual como espacios multidimensionales en los que las competencias de pensamiento computacional se median a través de la experiencia del usuario con referencia a la programación, involucrando el aprendizaje del usuario, la tarea de programación en lenguaje en que se hace la implementación y la herramienta en la que se trabaja que normalmente, es Scratch.

Por tanto, la tendencia hacia los trabajos futuros con respecto a la ingeniería de método para lograr una mezcla efectiva de estrategias que permitan el mejoramiento de las competencias de abstracción de los estudiantes a través de un entorno visual de programación basada en bloques como Scratch, es la incorporación de la filosofía ágil, en énfasis en lo colaborativo, la ramificación y por supuesto en los diferentes aspectos de la resolución problemas y los mecanismos de abstracción.

En particular, se deberán promover investigaciones en las que haya experimentos controlados que permitan la evaluación de las pautas de mejoramiento integral de los procesos de organización y división del trabajo en ChildProgramming pisando los mecanismos de abstracción y como método de evaluación la estadística descriptiva.

Igualmente, se promoverá el desarrollo de un kit de modelado con los niños enfatizando en el refinamiento del proceso y en lenguaje de desarrollo para facilitar el aprendizaje de los conceptos

básicos de la programación basada en bloques y la adquisición de competencias de pensamiento computacional para la resolución de problemas y el despliegue de acciones abstractas.

9.2. Aportes

9.2.1 Ámbito Social

Con el presente trabajo investigativo, se pretende aportar a aquellas iniciativas orientadas a que los niños puedan tener una gran participación e importancia en la programación. Lo anterior, proponiendo prácticas que ayuden a los maestros a fomentar las capacidades de los niños, mejorando su desarrollo mental, social y académico. El interés en esta área del conocimiento, está orientado hacia las iniciativas donde los niños puedan tener una mayor inclusión en actividades como el desarrollo de software, ingeniería y áreas afines.

9.2.2 Ámbito Académico

Este proyecto está basado en la necesidad de mejorar y motivar el aprendizaje de la programación y áreas afines, por esta razón es necesario formular un catálogo de prácticas que permitan la enseñanza de programación en niños. Así los resultados de este método pueden servir de base para adelantar iniciativas en la región.

9.2.3 Ámbito De La Investigación

Dada la importancia de conocer la experiencia de los niños con el desarrollo de software, este trabajo de grado está enfocado en mejorar y motivar a los niños en el desarrollo de la industria de software y las ciencias de la computación. Con el objetivo primordial de ayudar en la calidad educativa y al mismo tiempo mejorar el sector tecnológico del país.

9.2.4 Ámbito De Desarrollo

Teniendo en cuenta las falencias existentes en las instituciones educativas, este trabajo busca mejorar las aptitudes del desarrollo computacional de nuestros niños, logrando así, un aporte en la educación y aumentar el gusto por las áreas de las TIC para estos pequeños que son el futuro de nuestro país.

9.3. Limitaciones

Debido a que el lenguaje y las actividades del método fueron evaluados en casos muy concretos,

se necesita realizar más casos que permitan la evaluación más amplia para que los resultados pueden generalizarse que los resultados no puedan generalizarse.

A pesar de ello, se evidencia que un método tiene la potencialidad para organizar el proyecto y ayudar a la división del trabajo en los equipos cuando se utiliza ChildProgramming.

10. Referencias

4th Iberoamerican Workshop, H.-C. 2018. (2019). *Human-Computer Interaction* (V. Agredo-Delgado & P. H. Ruiz, Eds.). Springer. https://doi.org/10.1007/978-3-030-05270-6

Abelson, H., & Kong, S.-C. (2018). Computational Thinking in the STEM Disciplines. In *Computational Thinking in the STEM Disciplines*. Springer. https://doi.org/10.1007/978-3-319-93566-9

Ahmed, U. Z., Christakis, M., Efremov, A., Fernandez, N., Ghosh, A., Roychoudhury, A., & Singla, A. (2020). Synthesizing tasks for block-based programming. *Advances in Neural Information Processing Systems*, 12.

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, *55*(7), 833–835. https://doi.org/10.1093/comjnl/bxs074

B. Henderson-Sellers. (2006). Method Engineering : Theory and Practice. *Information Systems Technology and Its Applications, 5th International Conference*, 13–23.

Ballard, D. H. (2015). Brain computation as hierarchical abstraction. In *Brain Computation as Hierarchical Abstraction*. The MIT Press. https://doi.org/10.7551/mitpress/9780262028615.001.0001

Basogain, X., Olabe, M., & Olabe, J. (2015). *Pensamiento Computacional a través de la Programación: Paradigma de Aprendizaje*. 34. https://doi.org/10.6018/red/46/6

Bautista, J. S., Granados, J., Ortiz, D., & Reinoso, L. J. (2009). *Representaciones Mentales del Pensamiento*. Pontificia Universidad Javeriana.

Beecher, K. (2017). *Computational Thinking*.

Berciano-Alcaraz, A., Salgado-Somoza, M., & Jiménez-Gestal, C. (2022). Alfabetización computacional en educación infantil:Dificultades y beneficios en el aula de 3 años. *Revista Electrónica Educare*, *26*(2), 1–21. https://doi.org/10.15359/ree.26-2.15

Boroditsky, L., Ramscar, M., & Frank, M. (2002). *The roles of body and mind in abstract thought*. *13*(2), 185–189. https://doi.org/10.1111/1467-9280.00434

Bourret, R. (2015). *An Introduction to Programming with Scratch*. MIT Press. https://doi.org/10.1016/B978-0-12-088559-6.X5000-6

Brinkkemper, S., Lyytinen, K., & Welke, R. J. (1996). Method engineering. *Internet and Information Technology in Modern Organizations: Challenges and Answers - Proceedings of the 5th International Business Information Management Association Conference, IBIMA 2005*, *1*, 331. https://doi.org/10.1145/142750.142822

Carnevale, A., Smith, N., & Melton, M. (2011). *Stem: Science Technology Engineering Mathematics*. Universidad de Georgetown. https://doi.org/10.1353/chapter.471759

Chimunja, A., Collazos, C., & Hurtado, J. (2017). *ChildProgramming-C: como una mejora de la dimensión colaborativa del modelo ChildProgramming*. 9.

CSTA. (2017). CSTA K-12 computer science standards. In *CSTA Annual Conference 2017* (pp. 1–43). ACM.

Damerow, P. (1996). *Abstraction and Representation*. Springer.

Dhariwal, S. (2018). *Scratch Memories*. MIT Press.

Fazi, B. (2018). *Contingent Computing*. Rowman & Littlefield International, Ltd.

Fronza, I., Ioini, N. el, Pahl, C., & Corral, L. (2019). Agile and Lean Concepts for Teaching and Learning. In *Agile and Lean Concepts for Teaching and Learning*. Springer Singapore. https://doi.org/10.1007/978-981-13-2751-3

Frorer, P., Hazzan, O., & Manes, M. (1997). *Revealing the faces of abstraction*. Kluwer Academic Publishers.

Gacitúa-Bustos, R. A. (2014). Métodos de desarrollo de software: El desafío pendiente de la estandarización. *Theoria*, *12*(June), 23–42.

Gentner, D., & Stevens, A. L. (1983). Mental Models. In *The American Journal of Psychology* (Vol. 97, Issue 3). Lawrece Erlbaum. https://doi.org/10.2307/1422536

Gunbatar, M., & Turan, B. (2019). The Effect of Block-Based Programming on the Computational Thinking Skills of Middle School Students. *The Turkish Online Journal of Educational Technology*, *2*, 5.

Hermans, F., & Aivaloglou, E. (2017). *Teaching Software Engineering Principles to K-12 Students: A MOOC on Scratch*. 10. https://doi.org/10.1109/ICSE-SEET.2017.13

Hu, Y., Chen, C. H., & Su, C. Y. (2021). Exploring the Effectiveness and Moderators of Block-Based Visual Programming on Student Learning: A Meta-Analysis. *Journal of Educational Computing Research*, *58*(8), 1467–1493. https://doi.org/10.1177/0735633120945935

Hurtado, J., Collazos, C., Cruz, T., & Rojas, O. (2011). *Child Programming: Una Estrategia de Aprendizaje y Construcción de Software Basada en la Lúdica, la Colaboración y la Agilidad*. 6.

Hurtado, J., Gómez, V., & Zambrano, A. (2017). *Estudiando el modelo ChildProgramming-G para encontrar elementos que permitan desarrollar un sistema de memoria transactiva*. *6*, 10.

Johnson-Laird, P. N. (1980). Mental Models in Cognitive Science. *Cognitive Science*, *4*, 71–115.

Johnson-Laird, P. N. (2004). *Mental models in cognitive science*. Princeton University Press.

Jones, N. A., Ross, H., Lynam, T., Perez, P., & Leitch, A. (2011). Mental models: An interdisciplinary synthesis of theory and methods. *Ecology and Society*, *16*(46), 13. https://doi.org/10.5751/ES-03802-160146

Keller, R. M. (2001). *Computer Science: Abstraction to Implementation*. Robert Keller.

Khine, M. S. (2018). Strategies for Developing Computacional Thinking. *Computational Thinking in the STEM Disciplines: Foundations and Research Highlights*, *March*, 1–325. https://doi.org/10.1007/978-3-319-93566-9

Kim, C. M., Yuan, J., Vasconcelos, L., Shin, M., & Hill, R. B. (2018). Debugging during block-based programming. *Instructional Science*, *46*(5), 767–787. https://doi.org/10.1007/s11251-018-9453-5

Kramer, J. (2007). Is abstraction the key to computing? *Communications of the ACM*, *50*(4), 37–42. https://doi.org/10.1145/1232743.1232745

Laanpere, M., & Kori, K. (2020). *Informatics in Schools: Engaging Learner in computational thinking: Vol. 12518 LNCS*. Springer. https://doi.org/10.1007/978-3-030-63212-0_2

Maida, E., & Pacienzia, J. (2015). Metodologías de desarrollo de software [Tesis de licenciatura,

Universidad Católica Argentina]. In *Biblioteca Digital de la Universidad Católica Argentina*. http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf

Manktelow, K., & Chung, M. C. (2004). *Psychology of Reasoning*. Psychology Press. https://doi.org/10.4324/9780203506936

Marion, J., & Richardson, T. (2023). *Managing Projects with PMBOK 7*. BEP.

Marji, M. (2014). *Learn to program with Scratch : a visual introduction to programming with games, art, science, and math*. No Starch Press.

McManus, S. (2009). Scratch Programming in easy steps. In *Communication of the acm* (p. 35). MIT Press.

Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies*, *23*(4), 1483–1500. https://doi.org/10.1007/s10639-017-9673-3

Orejuela, H., García, A., Hurtado, J., & Collazos, C. (2013). Analizando y aplicando la gamificación en el proceso ChildProgramming. *Revista Colombiana de Computación*, 17.

Pelánek, R., & Effenberger, T. (2022). Design and analysis of microworlds and puzzles for block-based programming. *Computer Science Education*, *32*(1), 66–104. https://doi.org/10.1080/08993408.2020.1832813

PMI. (2021). *Pmbok Guide* (7th ed.). PMI.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Reyes, K., & Saavedra, A. (2016). *YoungProgramming–Una estrategia, colaborativa y lúdica, en apoyo al Aprendizaje de la Programación en Equipos conformados por Jóvenes* (p. 112). Universidad del Cauca.

Ricarose, R. (2007). *OpenBlocks: an extendable framework for graphical block programming systems*. MIT.

Ricarose, R. (2016). *Family Creative Learning: Designing Structures to Engage Kids and Parents*

as Computational Creators Ricarose. MIT.

Rich, P., Egan, G., & Ellsworth, J. (2019). A framework for decomposition in computational thinking. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 7. https://doi.org/10.1145/3304221.3319793

Roscoe, J., Fearn, S., & Posey, E. (2014). *Teaching computational thinking by playing games and building robots*. *December 2014*, 5. https://doi.org/10.1109/iTAG.2014.15

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 34. https://doi.org/10.1007/s10664-008-9102-8

Saitta, L., & Zucker, J. (2013). *Lorenza Saitta Jean-Daniel Zucker*. Springer.

Senge, P. (1990). *La Quinta Disciplina:* Free Libros.

Serna, E. (2011). *La importancia de la abstracción en la informática*. 6.

Serna, E., & Polo, J. (2014). Lógica y abstracción en la formación de ingenieros: una relación necesaria. *Ingeniería Investigación y Tecnología*, 12.

Silamani, A., & Guirao, G. (2015). Utilidad y tipos de revisión de literatura. *Ene*, *9*(2), 0–0. https://doi.org/10.4321/s1988-348x2015000200002

Varela, F. J., Thompson, E., & Rosch, E. (1991). The embodied mind: Cognitive science and human experience. In *The Embodied Mind: Cognitive Science and Human Experience* (1st ed.). MIT Press. https://doi.org/10.29173/cmplct8718

Vlieg, E. A. (2016). Scratch by Example. In *Scratch by Example*. Springer. https://doi.org/10.1007/978-1-4842-1946-1

Wang, B. L. (2021). *Developing Resources for Debugging Education using Block-based Languages*. M.I.T.

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: Students' perceptions of blocks-based programming. *Proceedings of IDC 2015: The 14th International Conference on Interaction Design and Children*, 199–208. https://doi.org/10.1145/2771839.2771860

Weintrop, D., & Wilensky, U. (2017). *Comparing block-based and text-based programming in high school computer science classrooms*. 25. https://doi.org/10.1145/3089799

Williams, H. (2017). *No Fear Coding*. International Society for Technology in Education.

Wing, J. (2006). *Computational Thinking*. 3.

Woodcock, J. (2016). *Coding Games in Scratch* (Vol. 15, Issue 2). DK.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. Blekinge Institute of Technology. *ACM International Conference Proceeding Series*, 10. https://doi.org/10.1145/2601248.2601268

Wyeth, P., & Purchase, H. C. (2000). Programming without a computer: A new interface for children under eight. In *Proceedings - 1st Australasian User Interface Conference, AUIC 2000*. https://doi.org/10.1109/AUIC.2000.822080

Zamin, N., Ab Rahim, H., Savita, K. S., Bhattacharyya, E., Zaffar, M., & Katijah Mohd Jamil, S. N. (2018). Learning Block Programming using Scratch among School Children in Malaysia and Australia: An Exploratory Study. *2018 4th International Conference on Computer and Information Sciences: Revolutionising Digital Landscape for Sustainable Smart Society, ICCOINS 2018 - Proceedings*, 1–6. https://doi.org/10.1109/ICCOINS.2018.8510586

Zuñiga-Muñoz, R. F. (2016). *Explorando los Procesos de Abstracción Computacional en Niños, identificando el uso de los modelos mentales compartidos*. Universidad del Cauca.

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

TRABAJO DE GRADO – MODALIDAD TRABAJO DE INVESTIGACIÓN

ACTA DE TITULARIDAD DE DERECHOS DE PROPIEDAD INTELECTUAL PARA TRABAJOS DE GRADO, MODALIDAD INVESTIGACIÓN

El Trabajo de Grado titulado Block-Based Design: Un Método para el Diseño de Programasbasados en Bloques, cuyo objetivo es Proponer y evaluar un método para el diseño de programas basados en bloques, utilizando mecanismos de abstracción, propios del pensamiento computacional, para facilitar a los niños la construcción, descomposición e integración de nuevos bloques, el cual se desarrollará por el (los) estudiante(s) Javier Ricardo Muñoz Castillo, perteneciente(s) al programa de Ingeniería de Sistemas de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca y bajola dirección de Julio Ariel Hurtado Alegría. Docente adscrito(a) al Departamento de Ingeniería de Sistemas, se realizará como requisito para optar al título de Ingeniero de Sistemas. La idea original del proyecto es de Julio Ariel Hurtado Alegría identificado con C.C.76,317.623 de Popayán quien la presentó al mencionado Departamento y fue aceptada como tema para el proyecto de grado en referencia.

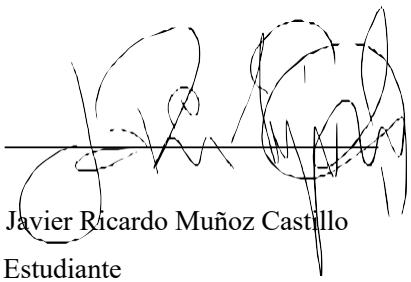
- Financiación:

Entidades que Participan en la financiación del proyecto	
1. Porcentaje Financiación UNIVERSIDAD DEL CAUCA	%100
2. Porcentaje Financiación entidades externas	% 0
3. Nombre de la(s) entidad(es) financiadora(s) (de ser el caso)	

Los resultados que surjan del desarrollo de este Trabajo de Grado estarán regidos por el Estatuto de Propiedad Intelectual de la Universidad del Cauca, el cual se encuentra reguladopor el Acuerdo 008 de 1999, modificado por el Acuerdo Superior 004 de 2018.

Todos los participantes manifiestan que conocen y aceptan lo dispuesto en el Acuerdo Superior No. 008 de 1999 y sus respectivas modificaciones, por lo cual, los aspectos no establecidos en la presente acta serán definidos de conformidad con el Estatuto de Propiedad intelectual de la Universidad del Cauca.

En constancia de aceptación, se firma el Acta por los que en ella intervienen, a los 3 días del mes de noviembre del 2022.



Javier Ricardo Muñoz Castillo

Estudiante

Cédula: 1.061.730.389

Código:104613020187

Nombre: Julio Ariel Hurtado Alegría

Director

Cédula: 76,317.623

Anexo A. Consentimiento informado

A QUIEN CORRESPONDA:

Por medio de la presente hago constar que yo, el Sr(a) Luz Dary Navarro Cardenás, identificado con la cedula de ciudadanía Nr° 34'329.672 de Popayán con domicilio en Popayán, Cl 19A # 4-04 como padre/madre del menor Emmanuel Ricardo Solarte Navarro identificado con tarjeta de identidad Nr° 1018458772 de 11 años de edad, inscrito en el **CURSO DE PROGRAMACIÓN DE SCRATCH PARA NIÑOS Y NIÑAS,** doy mi consentimiento para que asista de manera presencial y participe en el presente proyecto de investigación desarrollado por el grupo de investigación **IDIS de la Universidad del Cauca**, El curso se impartirá en las fechas 11, 18 y 25 de junio, 2 y 9 de julio del año 2022. Por lo que asumo toda responsabilidad ante cualquier suceso que pueda presentarse.

FIRMO BAJO CONFORMIDAD.

Anexo B. Observación de campo