SILE-AT: ENTORNO DE APRENDIZAJE SITUADO E INMERSIVO (SITUATED AND IMMERSIVE LEARNING ENVIRONMENT) PARA DESARROLLO DE PENSAMIENTO ALGORÍTMICO

JAVIER ALEJANDRO JIMÉNEZ TOLEDO

Tesis de Doctorado en Ciencias de la Electrónica

Director:
PhD. César A. Collazos
Doctor en Ciencias, mención Computación
Universidad del Cauca (Colombia)

Co-Director:
PhD. Manuel Ortega Cantero
Doctor en Ciencias
Universidad Castilla-La Mancha (España)

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Popayán, febrero 2022

JAVIER ALEJANDRO JIMÉNEZ TOLEDO

SILE-AT: ENTORNO DE APRENDIZAJE SITUADO E INMERSIVO (SITUATED AND IMMERSIVE LEARNING ENVIRONMENT) PARA DESARROLLO DE PENSAMIENTO ALGORÍTMICO

Tesis presentada a la Facultad de Ingeniería
Electrónica y Telecomunicaciones de la
Universidad del Cauca para la obtención del Título de

Doctor en:
Ciencias de la Electrónica

Director:
PhD. César A. Collazos
Doctor en Ciencias, mención Computación
Universidad del Cauca (Colombia)

Co-Director:
PhD. Manuel Ortega Cantero
Doctor en Ciencias
Universidad Castilla-La Mancha (España)

Popayán, Colombia
2022

Página de aceptación

A mis amados hijos: mi princesita Saray y Davis,
como una mínima recompensa por el tiempo
dejado de compartir durante estos años de estudio.
Gracias por permitirme estar a su lado para disfrutar lo maravilloso de la vida.

A mi hermosa esposa Ximena
Por ayudarme a salir de mis crisis,
preocupaciones y angustias
y por mantener vivo nuestro amor
pese a mi distanciamiento durante este largo tiempo,
siempre serás mi amor eterno!

A mi mamita Carmencita y mi papá Javier
por su apoyo incondicional
y por anteceder ante mi DIOS
en todas las situaciones de mi doctorado

A mis hermanos Helen y Robinson y mis sobrinos Santiago y Alejandro
por todo su apoyo y cariño

¡Gracias mi linda familia por vivir junto a mí, los logros y angustias en esta travesía!

Agradecimientos

A DIOS por permitirme ser su hijo.
Con mucho cariño, a mi querida Universidad CESMAG, quién me apoyó como becario de esta fabulosa experiencia.
Con aprecio al PhD. César Collazos, por su amistad, su comprensión, su liderazgo y por su enorme compromiso con mi formación, que sin importar la hora o el día estuvo siempre dispuesto a colaborándome. Gracias por cada una de sus valiosas enseñanzas de vida, que solo se viven y se aprenden cerca de su brillante actividad.
Con aprecio al PhD Manuel Ortega por su apoyo, amistad y enseñanzas en tierras lejanas.
Y finamente a todos los amigos y compañeros de estudio por su acompañamiento, lealtad y amistad durante esta bonita experiencia de vida.

Resumen Estructurado

Debido a las dificultades existentes en la formación académica de los constructores de software, se exploran inicialmente los problemas reportados en la literatura científica tanto en los procesos de enseñanza por parte de los profesores en los cursos de programación, como en los procesos de aprendizaje de los estudiantes. Luego, se establece que dicha literatura científica identifica que es en el primer curso de programación de computadores para constructores de software en formación (CS1), donde se concentra la mayor parte de las situaciones problemáticas, por ello, en un segundo momento se caracterizan tanto las metodologías existentes para abordar procesos algorítmicos en un CS1 como las herramientas existentes.

Analizados estos dos momentos, se encuentra que la mayoría de autores consultados focalizan el problema en la habilidad algorítmica computacional, dada principalmente por la inexperiencia previa del estudiante al enfrentarse a los diversos conceptos abstractos y que la dificultad referida se presenta justo en el momento de encuentro entre el estudiante y el profesor para abordar un concepto fundamental de programación en el aula de clase. Teniendo en cuenta esto, en un tercer momento se exploran estudios que utilizan la experiencia vivencial de los estudiantes para la asimilación de conceptos abstractos, encontrando que el modelo didáctico analógico es uno de los más utilizados en muchas áreas del conocimiento humano, pero la literatura reporta pocas experiencias de este modelo para la enseñanza o aprendizaje de un primer curso de programación con estudiantes universitarios.

Por lo anterior, se realiza este estudio que tiene como objetivo construir un entorno de enseñanza basado en analogías para desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios, el cual se lleva a efecto en tres grandes fases: en la primera se construye el marco conceptual necesario para soportar el presente estudio. En una segunda fase se diseña una propuesta metodológica basada en analogías y herramienta de visualización para el desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios. En la tercera fase se valida el entorno con la participación de estudiantes y profesores bajo el paradigma positivista, con enfoque cuantitativo, utilizando el método empírico analítico, con un tipo de investigación descriptivo y

mediante un diseño experimental con grupos de control y experimentales con post prueba.

Dicho entorno está conformado por dos modelos para el desarrollo de pensamiento algorítmico: el primero es un modelo de descubrimiento de conocimiento que utiliza el modelo didáctico analógico para la enseñanza del concepto fundamental de programación y el segundo es un modelo para la enseñanza del ejemplo computacional, el cual está basado en los proceso de construcción de la Ingeniería de Software a través de una herramienta de visualización.

Los principales resultados indican que para diseñar una analogía efectiva para el desarrollo de pensamiento algorítmico en un CS1 se deben combinar elementos morfosintácticos con inclusión de palabras claves propias de la terminología de ejemplos computacionales a nivel de verbos en infinitivo, en contextos cuantitativos y acompañados de sustantivos comunes y con frases neutrales o positivas.

Asimismo, se pudo establecer que la enseñanza de pensamiento algorítmico en potenciales programadores de computadores en un CS1 debe tomar como base los componentes estructurales del proceso de desarrollo de software para obtener resultados de aprendizaje evolutivos, ya que, desde este nivel, el estudiante debe comprender que la construcción de software obedece a una colección de actividades planificadas y bien diferenciadas.

Además, el entorno de enseñanza permite que el estudiante centre su atención en comprender en primer lugar el problema a solucionar a través de sus elementos esenciales y a la vez intuir desde un inicio la interfaz computacional a construir y sus respectivos diagramas de diseño y codificación.

Palabras claves: Entorno de enseñanza, Analogías, Pensamiento Algorítmico, CS1.

Structured Abstract

Due to the existing difficulties in the academic training of software builders, the problems reported in the scientific literature are initially explored both in the teaching processes by teachers in programming courses, and in the learning processes of students. Then, it is established that this scientific literature identifies that it is in the first course of computer programming for software builders in training (CS1), where most of the problematic situations are concentrated, therefore, in a second moment both the existing methodologies to address algorithmic processes in a CS1 and the existing tools are characterized.

Analyzed these two moments, it is found that the majority of authors consulted focus on the problem in computational algorithmic skill, given chiefly by the in previous student experience in confronting the various abstract concepts and that the difficulty referred is presented just at the moment of meeting between the student and the teacher to address a fundamental concept of programming in the classroom. Taking this into account, in a third moment studies that use the experiential experience of students for the assimilation of abstract concepts are explored, finding that the analog didactic model is one of the most used in many areas of human knowledge, but literature reports few experiences of this model for the teaching or learning of a first programming course with university students.

Therefore, this study is carried out that aims to instruct a teaching environment based on analogies for the development of algorithmic thinking in a CS1 for university students, which is carried out in three major phases: in the first one the conceptual framework necessary to support the present study is built. In a second phase, a methodological proposal based on analogies and visualization tool for the development of algorithmic thinking in a CS1 for university students is designed. In the third phase, the environment was validated with the participation of students and teachers under the positivist paradigm, with a quantitative approach, using the analytical empirical method, with a type of descriptive research and through an experimental design with control and experimental groups with post-test.

This environment is made up of two models: the first is a knowledge discovery model that uses the analog didactic model for the teaching of the fundamental concept of programming and the second is a model for the teaching of the computational example, which is based on the construction process of Software Engineering through a visualization tool.

The main results indicate that to design an effective analogy for the development of algorithmic thinking in a CS1, morphosyntactic elements must be combined with the inclusion of keywords typical of the terminology of computational examples at the level of verbs in infinitive, in quantitative contexts and accompanied by common nouns and with neutral or positive phrases.

Likewise, it was possible to establish that the teaching of algorithmic thinking in potential computer programmers in a CS1 must be based on the structural components of the software development process to obtain evolutionary learning results, since, from this level, the student must understand that the construction of software obeys a collection of planned and well-differentiated activities.

In addition, the teaching environment allows the student to focus their attention on first understanding the problem to be solved through its essential elements and at the same time intuiting from the beginning the computational interface to be built and their respective design and coding diagrams.

Keywords: Teaching environment, Analogies, Algorithmic Thinking, CS1.

Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Introducción

El pensamiento algorítmico es una habilidad que se debe desarrollar en todo constructor de software, desde su primer curso de programación de computadores (CS1) sin importar el paradigma de programación elegido para iniciar este proceso. Es por ello, que dicho CS1 es uno de los cursos más importantes en la formación de todo programador computacional, a tal magnitud que las habilidades y destrezas adquiridas en este por los estudiantes, determinarán el éxito o fracaso al bordar las temáticas de mayor complejidad en los cursos posteriores [1]. Por lo tanto, la mediación pedagógica que el profesor utiliza para lograr la asimilación de la lógica algorítmica en el estudiante, es de gran importancia para alcanzar resultados satisfactorios en su proceso de aprendizaje.

Es así como en esta investigación se propone un entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios. Para ello, se construyó un modelo de descubrimiento de conocimiento para analizar 570 experiencias de utilización de analogías en 15 universidades de 5 países de Centroamérica, Suramérica y Europa, con la participación de un total de 33 profesores expertos en CS1 en programas universitarios de computación que forman profesionales en construcción de software. Como resultado de este modelo, se propone una estructura para la construcción de analogías para la enseñanza del concepto fundamental de programación a través del modelo didáctico analógico, para luego complementar el entorno propuesto con la utilización de una propuesta metodológica para la enseñanza del ejemplo fundamental computacional a través de una herramienta de visualización.

El documento se encuentra estructurado de la siguiente forma: el capítulo 1 contiene los elementos claves del proyecto de investigación como: problema, pregunta, hipótesis, objetivos y metodología. En el capítulo 2 se encuentra el marco conceptual que soporta la investigación. El capítulo 3 exhibe el entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico en un CS1. En el capítulo 4 se presenta el resultado del proceso investigativo para la construcción de analogías mediante el modelo de descubrimiento de conocimiento y para el entorno de enseñanza

propuesto en esta investigación. En el capítulo 5 se finaliza con las conclusiones, los trabajos futuros y los productos construidos durante este estudio.

Capítulo 1

Proyecto de investigación

En la Figura 1 se presenta el diagrama jerárquico de las temáticas abordadas en este capítulo

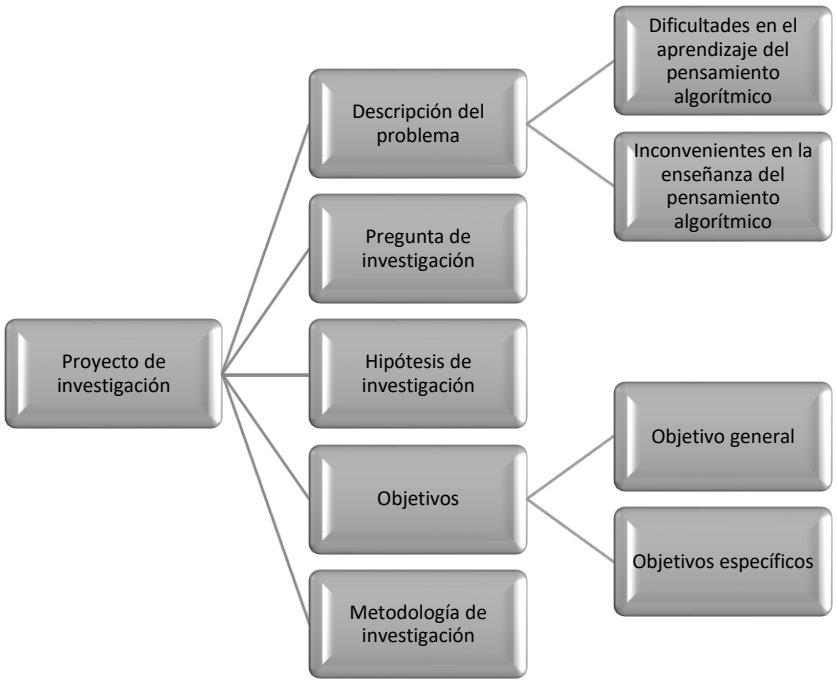


Figura 1. Diagrama jerárquico del Capítulo 1
Fuente: elaboración propia

1.1 Descripción del problema

El desarrollo de pensamiento algorítmico para constructores de software en un CS1, es un proceso complejo para muchos estudiantes noveles y al mismo tiempo se convierte en un desafío para los profesores [2], no sólo significa adquirir un conocimiento

nuevo por parte del estudiante, sino fundamentalmente, aplicarlo para resolver problemas [3]. Por ello, es una actividad cuidadosa que requiere que los estudiantes dominen habilidades cognitivas de orden superior tales como: abstraer información adecuada, desarrollar y aplicar modelos mentales o matemáticos, generar pseudocódigos y algoritmos, aprender sintaxis y semántica para codificación [4], entre otras. Además, para el proceso de enseñanza, los factores de motivación, la correcta armonía de conceptos y las didácticas aplicadas deben ser planeados y ejecutados adecuadamente por parte del profesor para minimizar el impacto de la abrumadora cantidad de conceptos y situaciones nuevas para un estudiante al abordar los fundamentos de programación en el aula de clase [5], que son la base primordial para el desarrollo de pensamiento algorítmico en la formación de constructores de software.

Asimismo, y a pesar del avance tecnológico del que somos testigos en la actualidad, existen diversos problemas derivados de la deficiente fundamentación en los procesos de aprendizaje tanto de los presentes como de los futuros profesionales de la industria del software y cuyos inconvenientes se originan desde el CS1, que generalmente fue abordado en su primer año de ingreso a la universidad o institución de formación [5]. Dichos problemas se generan por diversas circunstancias, entre ellas se encuentra el desconocimiento de conceptos fundamentales de programación, la falta de habilidades para modelar y construir programas computacionales y hasta la poca disciplina al momento de la construcción cognitiva requerida para enfrentar dichos conceptos [6].

La literatura científica reporta problemas tanto en el proceso de enseñanza guiada por los profesores como de aprendizaje por parte de los estudiantes en el desarrollo de pensamiento algorítmico al enfrentar el estudio de los fundamentos de programación [6]. Las principales dificultades en la enseñanza/aprendizaje se describen a continuación:

1.1.1 Dificultades en el aprendizaje del pensamiento algorítmico

La asimilación de los fundamentos de programación para el diseño de algoritmos básicos no es una tarea fácil para el estudiante debido a que involucra aspectos que van desde la motivación por aprender hasta el análisis de sus propios estilos de aprendizaje, el conocimiento de experiencia previas, la facilidad de interpretar conceptos nuevos, etc. [7] [8] [9] [10]. Además, algunos estudiantes no adquieren las habilidades básicas a pesar de culminar un CS1[5]. Es por esto que el desarrollo de pensamiento algorítmico para constructores de software es difícil de aprender; Informáticos, Pedagogos y Psicólogos llevan décadas investigando las dificultades que encuentran los alumnos para su aprendizaje [11].

A su vez, el proceso de aprendizaje requerido para el desarrollo de pensamiento algorítmico se considera como una tarea difícil ya que necesita que los estudiantes desarrollen destrezas del campo cognitivo para representar y resolver una situación problemática, acompañadas del estar dispuesto a aprender diversas sintaxis y semánticas requeridas para codificar programas de computadores [4], conllevando a que ciertos estudiantes sientan frustración y en casos extremos se retiren de sus programas de estudio [4][12][13].

Dann, Cooper y Pausch [14] determinan la existencia de cuatro elementos que complican el aprendizaje de la algoritmia computacional: metodologías de enseñanza inapropiadas para el tratamiento de la sintaxis en la codificación, el no tener el resultado tanto de los cálculos como el seguimiento de la estructura sintáctica al mismo tiempo cuando se ejecuta un código, la dificultad en la asimilación de lógica computacional y finalmente, la inapropiada utilización de los procesos de diseño algorítmico.

A nivel mundial se han llevado a cabo investigaciones que reportan carencias frente al desarrollo de pensamiento algorítmico en las etapas de formación para constructores de software, pero son escasos los estudios que presentan de forma clara y contundente el fundamento científico de la estructura de esta destreza [15] y a pesar que en la actualidad existen metodologías, enfoques, métodos y herramientas de enseñanza-aprendizaje, no existe una solución que satisfaga todos las necesidades por consenso[16][7] y que logre cumplir con éxito las necesidades de los estudiantes del siglo XXI [17].

Baldwin y Kulijis [18][5] identifican otro factor que obstaculiza en los estudiantes el desarrollo del pensamiento algorítmico, el cual está asociado a los complejos procesos cognoscitivos requeridos en las primeras etapas de estudio, ya que estos involucran tareas de orden superior del pensamiento como lo son la planificación, el razonamiento y la misma solución de problemas. Por ello, la importancia de desarrollar ciertas habilidades de pensamiento antes de enfrentar un CS1 se convierte en un factor decisivo en el correcto aprendizaje de la lógica algorítmica [5]. Además, en este momento se cuenta con muchos recursos didácticos disponibles para el aprendizaje tanto de la lógica algorítmica como también para su codificación, pero a pesar de ello, aún no existen entornos completos ni tampoco metodologías precisas que conlleven a un aprendizaje efectivo teniendo en cuenta las características individuales de los estudiantes [19].

Otra dificultad de los estudiantes para el desarrollo de pensamiento algorítmico al enfrentar un CS1, es el manejo de una terminología totalmente desconocida en su entorno experiencia, como lo es el caso de la concepción de “variable”, el manejo de

memoria asignado a un estado o simplemente el concepto de “tipado de datos” que hacen aún más confuso el aprendizaje en esta primera etapa [20]. En el mismo sentido, algunos autores también argumentan que la complejidad presentada en la estructura sintáctica del código, también dificulta los procesos de aprendizaje en el estudiantado como también la mala calidad de los instrumentos de aprendizaje existentes y el débil desarrollo de destrezas necesarias al afrontar un problema [5]. Asimismo, los estudiantes encuentran inconvenientes en la interpretación de los enunciados planteados, que pueden ser consecuencias de problemas relacionados con la deficiencia en procesos de abstracción [21]. Por ello, algunos estudiantes de estos CS1 que no logran alcanzar las habilidades necesarias para el desarrollo de su pensamiento algorítmico, adquieren una actitud de rechazo ante el proceso de aprendizaje [22][23][24][5].

Aparte de los problemas mencionados, tanto universidades como instituciones educativas aún siguen discutiendo sobre cuál debe ser el paradigma de programación apropiado para un CS1, discusión centrada principalmente en los de mayor utilización como el paradigma Estructurado, Orientada a Objetos (POO), Orientado a Eventos o Funcional [25]. El proceso de aprendizaje en cada uno de estos paradigmas trae consigo un sinnúmero de inconvenientes, por ejemplo, el problema del aprendizaje de la POO radica en que requiere la integración de varios elementos como lo es la asimilación del paradigma orientado a objetos, el manejo de un entorno de desarrollo y su correspondiente lenguaje de programación, la incorporación de una metodología adecuada de construcción, el dominio del mismo lenguaje unificado de modelado, la concepción de patrones y la lógica necesaria de abstracción para finalmente convertirla en código de programación [26]. Por esto y muchas cosas más, el estudiante se enfrenta en un periodo de tiempo corto ante una cantidad de nuevos y extraños conceptos que terminan haciendo más compleja la adopción de los concepciones necesarios para la formalización de sus lógica algorítmica que finalmente conlleva a la construcción de programas computacionales [27].

Además, otra de las dificultades para el aprendizaje de la POO lo constituyen las constantes actualizaciones de los entornos de programación por su alto contenido profesional que contiene una amplia disponibilidad de herramientas que resultan desconcertantes para un estudiante novato [26].Así mismo, los métodos de estudio inadecuados se convierten en un gran problema al momento de consolidar la lógica de pensamiento y el desarrollo de habilidades cognitivas requeridas para el desarrollo de pensamiento algorítmico [28].

En la Tabla 1 se presentan los principales hallazgos relacionados con las dificultades para el aprendizaje del pensamiento que enfrentan los estudiantes y el análisis de sus principales causas y efectos.

Dificultades	Posibles causas	Principal efecto
Motivación por el aprendizaje	Complejidad en las temáticas	Baja asimilación de conceptos Bajo rendimiento académico
Métodos de estudio inapropiados	Desconocimiento de los métodos de estudio	Aprendizaje memorístico que no aporta al desarrollo del pensamiento lógico
Bajos niveles de abstracción	Falta de experiencia o preparación de su modelo mental	No relaciona los conceptos con su entorno experiencial
Escasas habilidades de pensamiento algorítmico	Bajo desarrollo del pensamiento lógico No existencia de procesos de selección acordes con perfiles de aspirantes	Limitación para desarrollar procesos algorítmicos
Debilidad para la resolución de problemas	Débil formación al respecto	Poca capacidad de generación de alternativas para resolver problemas
Bajos niveles de desarrollo de pensamiento matemático	Problemas de comprensión matemática Bajos niveles académicos en el componente matemático	Dificultad en la solución de problemas
Débil proceso de conocimiento procedimental	Falta de preparación en los niveles educativos anteriores	Dificultad para construir algoritmos
Deficiente comprensión de la metodología de enseñanza del profesor	Problemas didácticos en la metodología de enseñanza del profesor	Débil asimilación de los conceptos enseñados

Dificultades	Posibles causas	Principal efecto
Escasa comprobación de resultados de manera instantánea	Construcción de algoritmos, procedimientos y demás en niveles de representación conceptual como por ejemplo los ejercicios en papel o tablero	Baja motivación, confusión
Cantidad de conceptos nuevos para asimilar	Curos de corta duración para un CS1	Confusión en la implementación de conceptos
Estudiar con recursos de aprendizaje inapropiados	Recursos didácticos que no contemplan los diversos estilos de aprendizaje	Limitación en el autoaprendizaje
No comprensión de enunciados a resolver	Bajos niveles de comprensión lectora Inapropiada redacción de enunciados por parte del profesor	La solución presentada (si la logra diseñar) dista de la necesidad planteada
Inadecuada articulación de temáticas	No existe un verdadero consenso de la forma de abordar un CS1	Choque de estructuras mentales del estudiante conforme avanzan los diversos cursos de programación

Tabla 1. Dificultades, causas y efectos del aprendizaje de la programación en CS1.
Fuente: elaboración propia

En la actualidad, los estudiantes continúan enfrentando dificultades para desarrollar su pensamiento algorítmico en un CS1 y los estudiantes parecen estar menos interesados en la programación, aunque su familiarización con las computadoras sugeriría que su aprendizaje sería más fácil hoy en día [29][17].

1.1.2 Inconvenientes en la enseñanza del pensamiento algorítmico

La literatura científica reporta varios inconvenientes respecto a la enseñanza del pensamiento algorítmico para constructores de software en formación, comenzando por la incertidumbre que se tiene al abordar un CS1 en lo relacionado con la secuencia de los saberes como «object-first» o «procedural-first», y aunque existen algunos estudios,

aún se hace complejo tomar una decisión soportada totalmente con elementos de juicio que determinen la mejor alternativa [30].

A su vez, la enseñanza de la lógica algorítmica establecen un reto para el profesor, que aparte del proceso metodológico, debe poner a prueba la acción didáctica con el propósito de contar con recursos significativos que guíen al estudiante de manera adecuada en la asimilación de los conceptos necesarios, por ello, es una tarea compleja teniendo en cuenta que son estudiantes novatos y muchos de ellos sin experiencia alguna en algoritmia [25].

Infortunadamente, los mecanismos de instrucción en la iniciación a la algoritmia computacional tienen un proceso limitado de acuerdo con los avances científicos y tecnológicos vivenciados, y a pesar de la aceptación de la existencia de la problemática por parte de la comunidad académica y científica, las metodologías desarrolladas en las aulas de clase aún contemplan los mecanismos de la vieja escuela basados en modelos por imitación y caracterizados por la exposición que hace el profesor ante la solución de un determinado problema, con la esperanza de que el estudiante lo implemente en su modelo mental [31].

En consecuencia, es la enseñanza de la algoritmia computacional el área donde se concentra la mayor preocupación de acuerdo al reporte científico publicado en la literatura [32]. Es por ello, que los escritos existentes afirman que en la mayor parte son los profesores quienes no inducen procedimientos adecuados que conlleven a la búsqueda clara de solución de problemas a sus estudiantes, sin contar con técnicas de autorregulación del conocimiento que beneficien los procesos meta-cognitivos como la planeación, el control y la evaluación del estudiantado, además, sin proponer mecanismos introspectivos de evaluación y cuyo aprendizaje finalmente se reduce únicamente a la explicación brindada en ese momento por parte del profesor quien acude al planteamiento de situaciones demostrativas que no evidencian el proceso completo cognitivo requerido para que el estudiante logre asimilar el conocimiento [33].

Dentro de las metodologías existentes para enseñar la algoritmia computacional, existe una que consiste en hacer que el estudiante resuelva una cantidad importante de ejemplos y situaciones problemáticas con el propósito de realizar codificación en un lenguaje de programación, probar sus diseños y corregirlos hasta que el ejercicio quede completamente correcto, lo cual muchas veces genera que el estudiante dedique mucho tiempo a resolver problemas sintácticos del lenguaje de programación y en ocasiones a establecer una disputa absurda con el computador, realizando acciones preocupadas por

sobrepasar dicho error en lugar de establecer un procedimiento lógico enfocado a la solución del problema planteado y no al manejo del lenguaje [34].

Por otro lado, la enseñanza de la algoritmia computacional desde hace mucho tiempo se ha convertido en un desafío tanto para los mismos profesores como para profesionales de la educación, debido a que conlleva el desarrollar destrezas para resolver problemas, el construir algoritmos que modelen las soluciones planteadas, el determinar la validez de dichas soluciones al dominio, y hasta el tener el conocimiento de uno o más lenguajes de programación, y al mismo tiempo teniendo en cuenta de que se cuenta con estudiantes de un CS1 que ingresan por primera vez a un centro de formación profesional o tecnológico [25].

En la Tabla 2 se presentan los inconvenientes de mayor reporte en la literatura científica relacionados con los procesos de enseñanza para el desarrollo de pensamiento algorítmico computacional por parte de los profesores, junto con el análisis de sus principales causas y efectos.

Inconvenientes	Posibles causas	Principal efecto
Falta de consenso para un CS1	Profesor sin experiencia Actualización de currículos	Desaprovechar el desarrollo de habilidades fundamentales en un CS1
No recomendación de métodos de estudio	Profesores noveles Baja importancia del proceso académico de los estudiantes por parte del profesor	Baja motivación del estudiante que finalmente desfavorece el proceso de aprendizaje del estudiante
Didácticas inapropiadas	Metodologías basadas en modelos clásicos fundamentados en la repetición	El estudiante no logra adquirir los fundamentos de programación requeridos
Utilización de recursos de aprendizaje inapropiados	Recursos didácticos que no contemplan los diversos estilos de aprendizaje	Limitación en el proceso de aprendizaje por parte del estudiante

Tabla 2. Inconvenientes, causas y efectos de la enseñanza de la programación en un CS1.
Fuente: elaboración propia

Frente a la problemática para el desarrollo de pensamiento algorítmico aquí analizada, es posible evidenciar que la mayor parte de esta se presenta justo en el momento de encuentro entre el estudiante y el profesor para abordar un concepto fundamental de programación en el aula de clase y es en ese instante cuando se coloca en juego la abstracción por parte del estudiante y la forma como el profesor transmite dicho conocimiento. Por lo tanto, es en ese momento donde existe la ruptura en la asimilación de los conceptos [1], por ello, y teniendo en cuenta que una buena parte de los estudiantes no han tenido experiencia previa en este campo [7] [8] [9] [10], es una responsabilidad directa del profesor el ofrecer estrategias didácticas que acerquen de una manera vivencial al estudiante mediante procesos de abstracción cómodos a las experiencias y presaberes incorporados en su diario vivir [1][5][14] [18] [20].

En lo relacionado con las dificultades del proceso de aprendizaje de la algoritmia computacional, la mayoría de autores consultados focalizan el problema principalmente en la experiencia previa del estudiante al enfrentarse a los diversos conceptos abstractos (que a primera vista no tienen para él equivalencia en la vida real) en un CS1 [7] [8] [9] [10] [27]. Sin embargo, también toman gran importancia inconvenientes relacionados con los estilos de aprendizaje, limitaciones en su capacidad de abstracción, cantidad de tiempo escaso para asimilar cantidades enormes de conceptos nuevos, y los continuos cambios en los entornos de programación.

1.2 Pregunta de investigación

¿La utilización de un entorno de enseñanza basado en analogías mejora el desarrollo de pensamiento algorítmico en estudiantes universitarios en un CS1?

1.3 Hipótesis de investigación

Acorde con la pregunta de investigación se plantearon las siguientes hipótesis:

Hi: La utilización de un entorno de enseñanza basado en analogías, es una estrategia que mejora el desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios, con relación a otras estrategias utilizadas en estos cursos.

Ho: La utilización de un en entorno de enseñanza basado en analogías es una estrategia que NO mejora los el desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios, con relación a otras estrategias utilizadas en estos cursos.

Ha: La utilización de un en entorno de enseñanza basado en analogías es una estrategia que mantiene el desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios, con relación a otras estrategias utilizadas en un estos cursos.

1.4 Objetivos

1.4.1 Objetivo general

Construir un entorno de enseñanza basado en analogías para desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios.

1.4.2 Objetivos específicos

Caracterizar los procesos y herramientas utilizadas para el desarrollo de pensamiento algorítmico en instituciones de educación superior a nivel de pregrado.

Diseñar una propuesta metodológica basada en analogías y herramienta de visualización para el desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios.

Validar el entorno con profesores y estudiantes de un CS1 utilizando grupos de control y experimentales para comprobar las hipótesis planteadas.

1.5 Metodología de la investigación

El proceso metodológico del presente estudio se realizó en tres etapas: etapa de caracterización, etapa de diseño y etapa de validación.

1.5.1 Etapa de caracterización.

En esta etapa, se realizó una caracterización de los procesos y herramientas utilizadas para el desarrollo de pensamiento algorítmico en instituciones de educación superior a nivel de pregrado. Para ello, se realizó una revisión sistemática de literatura utilizando un enfoque de procesos investigativos de la Ingeniería de Software, con el propósito de proporcionar una visión general de un área de investigación, e identificar la cantidad y el tipo de investigaciones y los resultados disponibles dentro de ella y así construir un esquema de clasificación y una estructura con validez.

En la revisión sistemática se plantearon las siguientes preguntas de investigación: RQ1: ¿Qué problemas de aprendizaje de la algorirtmia existen en los estudiantes de un CS1?, RQ2: ¿Qué inconvenientes existen en la enseñanza de la algorítmia en un CS1?,

RQ3: ¿Qué herramientas tecnológicas de enseñanza-aprendizaje algorítmicos se utilizan en un CS1?, RQ4: ¿Qué estrategias de enseñanza-aprendizaje de algoritmos se aplican en un CS1?, RQ5: ¿Existen algunas consideraciones metodológicas para enfrentar un CS1? y RQ6: ¿Qué tendencias tiene la programación de computadores?.

En dicha revisión sistemática re realizó una exploración en cuatro bases de datos de referencias bibliográficas de publicaciones científicas (Scopus, IEEE Xplorer, Springer, ISI Web of Science) en las cuales se obtuvo 106 estudios divulgados en los últimos siete años. Posterior a un proceso de selección y de evaluación de calidad, se determina que 46 coinciden con los criterios de la revisión, obteniendo como resultado la recopilación de las principales experiencias y prácticas reportadas en el proceso de enseñanza-aprendizaje de la programación de computadores. Además, la revisión sistemática permitió determinar las problemáticas asociadas, mediante la cual se fundamentó la problematoica planteada en el numeral 1.1 de este documento. Asimismo, se obtuvo un listado de 33 herramientas de software, 36 estrategias de trabajo, 18 consideraciones metodológicas, importantes recomendaciones y las tendencias futuras para afrontar un primer curso de programación de computadores, cuyos resultados se encuentran en el Capítulo 2 de este documento.

1.5.2 Etapa de diseño.

En esta etapa se diseñó un entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico para estudiantes universitarios. Dicho entorno combina elementos del modelo didáctico analógico (para acercar vivencialmente al estudiante al concepto a enseñar) con una herramienta de visualización (cuyo propósito es el desarrollo de ejemplos computacionales). Además, el entorno presenta unas habilidades que se sugiere debe poseer los estudiantes e incluye recomendaciones para los profesores encargados de orientar un CS1.

Los resultados de esta etapa se presentan en el Capítulo 3 de este documento.

1.5.3 Etapa de validación.

El proceso de evaluación de este estudio se realizó en dos momentos, en primer lugar, se evaluó la construcción de analogías únicamente con profesores de un CS1 y luego con los estudiantes de dichos profesores. Para el primer momento de evaluación, se utilizó el paradigma positivista, con enfoque cuantitativo, utilizando el método empírico analítico, con un tipo de investigación descriptivo y mediante un pre-experimento con pre prueba y post prueba.

En este proceso experimental participaron dos profesores de dos universidades en Colombia con un diseño experimental basado en G O_1 X O_2, donde G fue el grupo experimental conformado por los profesores que orientan los CS1. Asimismo, O_1 fue conformado por el banco de ejemplos de analogías que a dichos profesores se les pidió que redactaran para explicar los conceptos de Entrada, Salida, Condicional y Ciclos, presentando un total de 14 ejemplos para el curso Introducción a la Programación y 17 ejemplos para Fundamentos de Programación.

Luego, a los profesores se le suministró el tratamiento experimental X que consistió en el modelo de creación de analogías propuesto en esta investigación. Finalmente, se les pidió que construyeran nuevamente las analogías por cada concepto, de acuerdo a las recomendaciones del tratamiento experimental y cuyo resultado fue O_2, obteniendo el mismo número de ejemplos iniciales planteados por cada profesor.

En un segundo momento, se evaluó la implementación completa del entorno de enseñanza utilizando las analogías de acuerdo al banco de ejemplos obtenidas en O_2 con estudiantes de un CS1 y utilizando a la vez CodES. Los resultados obtenidos se compararon con los cursos anteriores orientados por los mismos profesores en los que utilizaron como metodología las clases magistrales expositivas combinadas en algunas ocasiones con las analogías recolectadas en O_1. Es este caso, el experimento tuvo el mismo paradigma, enfoque, método y tipo de investigación ya mencionados con los profesores, con la diferencia que el diseño fue cuasi experimental con grupos de control y experimentales con pre y post prueba. En el proceso experimental participaron 154 estudiantes distribuidos en 4 grupos y cuyo diseño experimental fue:

$$G_1\ O_3\ X\ O_4$$

$$G_2\ O_5 – O_6$$

$$G_3\ O_7\ X\ O_8$$

$$G_4\ O_9 – O_{10.}$$

En definitiva, el proceso investigativo se desarrolló bajo el paradigma positivista, debido a que se orientó desde una visión nomotética de la investigación, con un enfoque cuantitativo, porque que se implementaron técnicas estadísticas que permitieron comprobar las hipótesis formuladas, mediante un método empírico analítico con su rigurosidad establecida y con un diseño de investigación con preprueba, posprueba, grupos experimentales y de control a quienes se les aplicó un tratamiento experimental (X) que consistió en la propuesta metodológica basada en el entorno de enseñanza para el desarrollo de pensamiento algorítmico con el propósito de determinar la incidencia de dicho tratamiento.

Capítulo 2

Marco conceptual

En este capítulo se encuentran los fundamentos teóricos que soportan la investigación realizada, abordando principalmente el pensamiento algorítmico y las analogías. En la en la Figura 2 se presenta el mapa mental del marco conceptual.

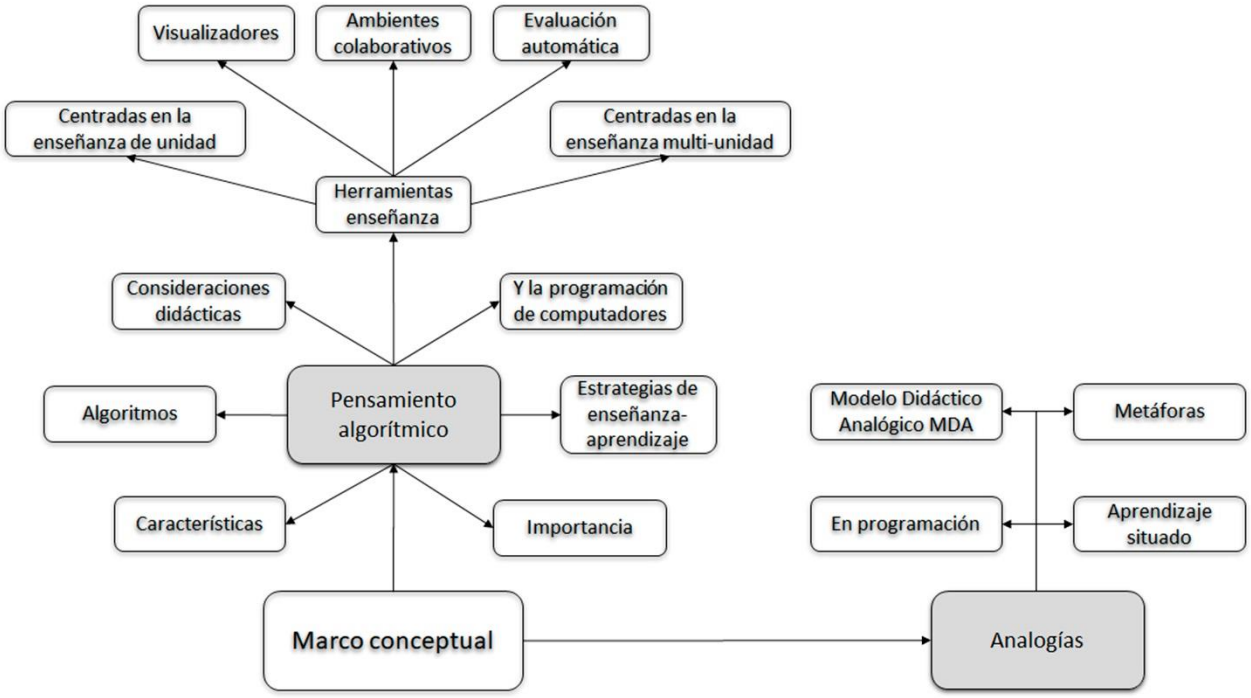


Figura 2. Mapa mental del marco conceptual
Fuente: elaboración propia

2.1 Pensamiento algorítmico

El pensamiento algorítmico disciplina, ordena y formaliza el pensamiento del estudiante [35] y su estudio en edades tempranas conlleva la preparación requerida en programas universitarios o tecnológicos como en áreas ingenieriles, matemáticas, físicas,

contables, entre otras [36]. Además, quienes lo estudian poseen varias competencias metacognitivas, creativas, de orientación y de pensamiento [37].

El desarrollar pensamiento algorítmico es un reto y para muchos estudiantes el obstáculo en un CS1, donde la apropiación y correcta utilización de la estructura de control y selección es lo más complejo de alcanzar [38]. Además, la gran mayoría de los estudiantes inscritos en un CS1 tienen escasas destrezas algorítmicas, y al finalizar el curso un pequeño porcentaje de ellos logran un nivel satisfactorio [39].

2.1.1 Consideraciones sobre algoritmos

En los estudios realizados por Knuth [40] el origen de la palabra algoritmo tiene dos posibilidades, en el primer postulado se establece que se origina de la palabra "Algorismo" propuesta por Al-Khowârizmî en su libro escrito en el año 825 y denominado "reglas de restauración y reducción" cuya definición (dada en 1957 por el diccionario Webster's New World Dictionary) lo establecía como un proceso aritmético usando guarismos arábigos [41]. El segundo postulado afirma que fue propuesta en el siglo IX por el matemático de origen Árabe llamado al-Jwârizmi quien definió "algoritmia" a las reglas de las cuatro operaciones aritméticas del sistema decimal, razón por la cual este término fue utilizado por matemáticos [41].

Futschek [42] propone el algoritmo como un método de resolución de problemas mediante instrucciones debidamente determinadas, donde no existe un algoritmo para diseñar algoritmos [43]. Además, un algoritmo es la colección de reglas, que no están sujetas a otras reglas [44], para realizar cálculos en forma manual o computarizada [45], que está conformado por una serie de pasos lógicos para desarrollar una tarea y su creación es en esencia realizada por seres humanos, además, se encuentran en una vasta parte de su actual accionar [46] e incluyen métodos de modelación y creatividad y no dependen de los lenguajes de programación ya que no necesariamente se deben computarizar [41] y se pueden manifestar como procedimientos matemáticos, líneas de código o datos y pueden expresarse de diferentes formas [47], por esto, un algoritmo es una representación formal y un enfoque creativo [35]. A su vez, los algoritmos de tipo informáticos concentran elementos invisibles y al mismo tiempo abstractos en su estructura diseñados para un entorno computacional [48] [49] y deben cumplir cinco condiciones: finitud, definibilidad, entradas, salidas y efectividad [40].

Según la Real Academia Española [50], la palabra algoritmo proviene posiblemente del latín "algobarismus" como abreviación del cálculo de cifras arábigas y hace referencia al "1.. Conjunto ordenado y finito de operaciones que permite hallar la

solución de un problema” y “2. Método y notación en las distintas formas del cálculo”.

2.1.2 Pensamiento algorítmico como base del pensamiento computacional

El pensamiento algorítmico es inherente y parte importante del pensamiento computacional, debido a que los algoritmos son el fundamento esencial del pensamiento computacional. El enfoque del pensamiento computacional establece que la automatización de sus soluciones solo son posibles mediante la ejecución del pensamiento algorítmico, es decir, una solución informática derivada del análisis de una situación mediante pensamiento computacional, será posible únicamente mediante la ejecución del pensamiento algorítmico existente en dicha situación [51].

Por su parte, el pensamiento computacional es una destreza necesaria para cualquier usuario en el siglo XXI [52] y propone un paradigma de enseñanza-aprendizaje con características cuantitativas, centrado en la intuición y que incorpora conocimiento mediante prueba y error [53] y se define como un enfoque desarrollado para solucionar problemas mediante el pensamiento crítico y el accionar computacional bajo herramientas de construcción algorítmica, o a través de lenguajes de programación, o con procesos de simulación e inteligencia artificial, etc. [54] [55] [56]. El pensamiento computacional es la base para el desarrollo de aplicaciones informáticas y es a la vez una herramienta para apoyar la solución de situaciones problemáticas en diversas áreas [57], aplicando conceptos básicos informáticos [58].

Wing [59] define que “el pensamiento computacional consiste en la resolución de problemas, el diseño de los sistemas, y la comprensión de la conducta humana haciendo uso de los conceptos fundamentales de la informática” (p. 34) y propone las siguientes características: formulación, organización lógica de los datos, abstracción, algoritmo, implementación y creación de nuevos algoritmos.

Hoy en día, la enseñanza de pensamiento computacional ha tomado gran importancia en escuelas primarias en todo el mundo [60], incorporándola de forma transversal en cursos como matemáticas, ciencias, arte y ciencias sociales, sin aumentar las horas de clase [61].

2.1.3 Características del pensamiento algorítmico

El pensamiento algorítmico tiene relación directa con el pensamiento computacional [62] [63][52] y es una habilidad que todas las personas deben tener [46], además, es una competencia que incorpora el pensamiento abstracto y lógico, el pensamiento en estructuras, la creatividad, etc., para resolver problemas [64],

permitiendo plantear una solución mediante una serie de pasos [46], logrando representar situaciones complejas mediante instrucciones simples [65].

A pesar del auge que toma el pensamiento algorítmico establecido por Grover en 2009, la National Academy of Sciences lo consideró nuevamente como pensamiento computacional en 2010, el cual hacía referencia a una destreza intelectual esencial como leer, escribir, narrar y realizar cálculos aritméticos [66].

Asimismo, el pensamiento algorítmico es un sistema de métodos mentales [67]y permite que el estudiante desarrolle su pensamiento, analice un contexto y diseñe una solución adecuada sin utilizar un lenguaje de programación [68], de esta manera el pensamiento algorítmico puede producir conocimiento conceptual [69] debido a que es una habilidad matemática mediante la cual es posible solucionar problemas, modelarlos algorítmicamente y automatizarlos [37]. Además, pretende que el estudiante construya un plan compuesto de una lista de pasos en el que se define el problema, se divide en unidades más pequeñas y determina una solución [70], es decir, proporciona un procedimiento para resolver un problema [52].

El pensamiento algorítmico estructura el pensar de forma lógica y organizada y mediante herramientas divide una situación compleja en una serie de acciones simples [71], además, provee la capacidad de ejecutar, validar, entender y construir algoritmos para afrontar diversas situaciones [72].

Analoca [41] propone la definición de pensamiento algorítmico desde la matemática mediante tres acciones: análisis del modelo, construcción de un diagrama de Flujo y su verificación, asimismo, plantea las siguientes fases para su enseñanza: adecuada formulación de situaciones programables, comprensión de ciclos y finalmente el manejo de restricciones (condicionales).

El pensamiento algorítmico también se concibe como una habilidad para construir, concebir, ejecutar y evaluar procesos computacionales [52] [73], entendiendo y ejecutando acciones paso a paso y que permitan la creación de nuevos algoritmos [46] con procesos de razonamiento y lógica en la escritura de las instrucciones requeridas para solucionar un problema [74]

A su vez, el pensamiento algorítmico forma parte de nuestra vida cotidiana, puesto que las diferentes actividades que realizan los humanos pueden establecerse como acciones algorítmicas [58], y hoy en día en la alfabetización informática este juega un papel importante debido a la utilización de algoritmos como elemento primordial de esta actividad [35].

Sarienko [35] describe que la base del pensamiento algorítmico está conformada por las cinco etapas de formación espiral de acciones mentales de Halperin [75]: 1. Partir

de un nivel básico, 2. Desarrollo de pensamiento operativo y figurativo, 3. Desarrollo de pensamiento verbal. 4. Desarrollo de pensamiento conceptual 5. Llegar a un nivel avanzado. Además, la literatura también resalta el apoyo de la teoría de la actividad en procesos de enseñanza de Talizina [76] para el desarrollo de pensamiento algorítmico mediante el cual el estudiante realiza etapas de planificación, ejecución y control en su proceso de aprendizaje [77].

Por su parte, Landa [78] propone una selección adecuada de situaciones para ser enfrentadas por los estudiantes en la formación del pensamiento algorítmico: uso de algoritmos de aprendizaje intencionados, utilización de algoritmos de manejo de elementos o acciones, empleo de métodos de pensamiento y desarrollo de la intuición.

Asimismo, Semenihina y Rudenko [36] hacen un acercamiento de las habilidades necesarias para la generación de pensamiento algorítmico desde sus experiencias escolares, proponiendo: el uso acciones importantes para el estudiante; incorporación de objetos de aprendizaje acordes a los posibles estilos de aprendizaje; motivación desde la acción psicológica; desarrollo de competencias mediante juegos; realización de actividades individuales y en equipo.

Malik y otros [79] proponen un enfoque basado en tres pasos que para el desarrollo de pensamiento algorítmico: Definición del problema, plan algorítmico de solución y generación de código.

También, Arkhipov [80] determina las siguientes tareas claves en la formación de pensamiento algorítmico: construcción de pensamiento lógico, estudio de la programación estructural, diseño de algoritmos y codificación en lenguaje de programación.

Por su lado, Altukhova y Smirnova [81] establecen que en la construcción del pensamiento algorítmico es necesario realizar en orden estas actividades: caracterizar procesos pedagógicos y psicológicos de los estudiantes; establecimiento de acciones secuenciales y construcción del modelo.

Al igual, Altukhova y Smirnova [81] realizan una descripción interesante de las habilidades que debe tener un ingeniero educador como formador de pensamiento algorítmico, entre ellas están: entender el postulado verdadero / falso; promover la agilidad mental; formulación de conclusiones; potenciar el pensamiento racional; asimilación de juicios obtenidos y reflexión del propio accionar.

Hromkovič y otros [82] proponen que los objetivos del plan de estudio del primer curso de programación deben ser los aspectos básicos de la generación de pensamiento algorítmico, entre los cuales están: el lenguaje formal, procesos de abstracción, capacidad de codificar y conocer los alcances de cómputo, además, afirma que el

pensamiento algorítmico es el elemento más importante de la computación donde la abstracción y la automatización son los conceptos primordiales.

Por su parte, Futschek [42] asimila el pensamiento algorítmico como el desarrollo de habilidades para la creación y comprensión algorítmica mediante la capacidad de: análisis de problemas, descripción precisa de la situación, solución básica, construcción algorítmica, dimensionar las posibles soluciones y optimar su eficiencia.

Finalmente, Christodoulou [43] afirma que el pensamiento algorítmico no se refiere únicamente a la facultad de construir algoritmos, sino que mediante este enfoque es posible obtener una mayor comprensión de un problema para posteriormente describirlo mediante una serie de pasos. Además, considera las siguientes habilidades: formulación, recolección, división, patrones, modelado, algoritmos, errores y comunicación.

2.1.4 Consideraciones didácticas del pensamiento algorítmico

La motivación juega un papel importante en el proceso de aprendizaje para iniciar y catalizar las habilidades requeridas en el desarrollo del pensamiento algorítmico [83] y una de las estrategia de aprendizaje más utilizadas es el reconocimiento de patrones, el cual se basa en realizar las siguientes acciones: escuchar, ver y hacer [46].

Por su parte, la enseñanza del pensamiento algorítmico trae consigo un conjunto de problemas didácticos, lo cual lo hace verdaderamente relevante [35], incluyendo una serie de consideraciones y principios que los estudiantes deben desarrollar como motivación continua, participación activa, reto progresivo y abstracción [46]. Además, es necesario desarrollar estructuras mentales como: secuencia, selección y repetición mediante pseudocódigo, diagramas de flujo y lenguaje de programación [41].

Para lograr en los estudiantes un estilo de pensamiento algorítmico desde la enseñanza, es esencial conocer tanto sus procesos de percepción psicológica como sus estilos de aprendizaje: visual, de audio y cinestésica [84]. Computacionalmente existen diversas herramientas para desarrollo de pensamiento algorítmico, entre ellas: pseudocódigo, bloques visuales, diagramas de flujo, diagramas de actividad, diagrama Nassi-Shneiderman, etc. [60]

Csernoch, Biró, Máth y Abari [39] determinan que solo se realiza desarrollo de pensamiento algorítmico en los cursos diseñados para tal fin y se desaprovecha la potencialidad algorítmica en muchos escenarios formativos que incluyen la informática donde solo se utilizan procesos de acercamiento superficial ineficientes.

En la enseñanza de la lógica algorítmica, la herramienta de mayor aceptación la constituyen los visualizadores de algoritmos [42] los cuales permiten “visualizar” uno o

varios seguimientos paso a paso de cada una de las instrucciones de un algoritmo de forma gráfica y/o mediante datos. Asimismo, la mayor parte de los primeros cursos de programación de computadores generalmente enseñan a construir algoritmos y al mismo tiempo lo combinan con el manejo de un lenguaje de programación [85]. Además, en la enseñanza del pensamiento algorítmico se deben construir tres habilidades: pensamiento creativo, raciocinio sistemático y trabajo colaborativo [86];

Por otro lado, entre los enfoques reportados para la enseñanza del pensamiento algorítmico en un CS1 se encuentra el enfoque constructivista como uno de los más utilizados, que se complementa con la teoría cognitiva de Piaget, en la cual, la construcción del conocimiento se realiza con procesos de asimilación para incorporar nuevas experiencias con los ya existentes en el sujeto y con procesos de acomodación que mediante patrones de entendimiento apropia un conocimiento desconocido.

También, otro enfoque apreciado en el mapeo sistemático de literatura para la enseñanza del pensamiento algorítmico es el construccionismo de Seymour Papert, el cual se apoya en la teoría cognitiva de Piaget y establece que una nueva idea es comparada y articulada con las estructuras mentales existentes en el sujeto resultado de sus conocimientos previos.

Malik, Shakir, Eldow y Ashfaque [79] proponen el modelo ADRI como un enfoque de enseñanza para la generación de pensamiento algorítmico que combina elementos de enfoque, despliegue, resultado y mejora.

A su vez, M. Zhaldak y Kurilov [87][88] proponen incorporar métodos de enseñanza activos para el desarrollo de pensamiento algorítmico como el aula invertida, el Aprendizaje Basado en Problemas o ABP y el Aprendizaje Colaborativo y Cooperativo; mientras que Minty [89] encuentra varias ventajas al utilizar la teoría para resolver problemas de forma inventiva; asimismo Semenihina y Rudenko relatan la importancia del enfoque metódico para la generación de pensamiento algorítmico [36].

Por otro lado, Lamagna [73] propone un enfoque desenchufado (sin utilizar herramientas computacionales) para la construcción de pensamiento algorítmico, tanto para estudiantes de ciencias computacionales como para aquellos de otras disciplinas, en el cual no es necesario el computador; Matyushchenko, Castelblanco, et al. [67] [90], plantean el desarrollo del pensamiento algorítmico por medio de aplicaciones robóticas; Mezak y Papak [91] diseñan escenarios de aprendizaje adecuados con situaciones cotidianas; Solomon [92] aborda la fuerza de la representaciones espaciales como elemento central para la enseñanza y el aprendizaje del pensamiento algorítmico en el aula de clase; por otra parte, Vinayakumar [52] et, al. utiliza la programación visual basada en bloques.

Teniendo en cuenta que la literatura científica reporta reiterativamente que el problema de la enseñanza del pensamiento algorítmico radica principalmente en la experiencia previa del estudiante al enfrentarse a los diversos conceptos abstractos y que la dificultad referida se presenta en el momento de encuentro entre el estudiante y el profesor para abordar un concepto en un CS1 [1] [5] [7] [8] [9] [10] [14] [27] [18] [20] [48] [49], en esta investigación se presenta una propuesta de un entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico en un CS1 con estudiantes universitarios.

Dicha propuesta utiliza una estrategia didáctica basada en el Modelo Didáctico Analógico y un modelo de descubrimiento de conocimiento para la construcción de analogías apropiadas, con el propósito de dar a conocer al estudiante únicamente el concepto fundamental de programación, con el fin de lograr una aproximación inicial de dicho concepto.

Una vez el estudiante tiene un acercamiento al concepto fundamental, se complementa con un modelo de enseñanza que incorpora las etapas del proceso de software para abordar ejemplos de modelado computacional a través de una herramienta de visualización algorítmica que se basa únicamente en el Diagrama de Entrada/Salida para facilitar la incorporación lógica procedimental. Por lo tanto, una vez el estudiante realice la abstracción del concepto fundamental y modele ejemplos computacionales, se realiza una serie de ejercicios de asimilación que terminan con una retroalimentación al proceso para fortalecer el desarrollo de pensamiento algorítmico en los estudiantes universitarios de un CS1.

2.1.5 El pensamiento algorítmico y la programación de computadores

El pensamiento algorítmico es un proceso inherente a la programación de computadores, ya que es la base fundamental que todo constructor de software debe poseer para obtener resultados adecuados tanto en producto final como en los procesos de flexibilidad y calidad que deben incorporar dichos productos.

Son muchos los beneficios de aprender a programar computadores, entre ellos permite el desarrollo de diversas competencias, como el pensamiento crítico, análisis de conceptos y resolución de problemas, además, los estudiantes aprenden a trabajar en grupos y colaborar entre ellos en su esfuerzo por desarrollar programas ejecutables mientras se ejercitan en el intercambio de conocimientos y la comunicación de ideas [17]. Por lo tanto, la programación de computadores capacita a los estudiantes para convertirse en aprendices de por vida, un beneficio muy importante para este mundo de

conocimiento en constante crecimiento, ya que pueden transferir sus habilidades a varios dominios de trabajo futuros [93].

La programación no es únicamente escribir código fuente ya que este es el resultado de una serie de actividades previas que le garantizan cualidades de ser flexible, robusto y acorde a los objetivos planteados, por ende, un programa informático es una colección de instrucciones que al ejecutarse efectúa actividades específicas a través de un sistema de cómputo y requiere para su escritura y ejecución de un lenguaje de programación que tiene una sintaxis con la cual fija las normas de codificación y una semántica que le permite plasmar sus objetivos en un entorno formal [94].

Programar computadores requiere mucha dedicación, lo cual se refleja en el tiempo invertido para esta actividad, además, en este proceso tanto el estudio de temáticas como el desarrollo de ejercicios puntuales no garantiza su efectivo aprendizaje, teniendo la necesidad de incorporar este nuevo modelo mental [95] que exige una constante actualización acorde con las tendencias experimentadas en los recientes enfoques emergentes de este campo.

Por otro lado, la programación de computadores permite resolver problemas puntuales que requieren de la mediación de objetos tecnológicos [96] con capacidades de procesamiento autónomo, almacenamiento interno o externo y que permitan brindar respuestas a través de los medios demandados, todo esto mediante metodologías y modelos específicos validados por toda una comunidad.

La resolución de problemas mediante programación requiere una serie de conocimientos y habilidades en campos como el modelado (lógica matemática y procedimental); la ingeniería con sus arquitecturas y procesos de software; y finamente la computación con sus algoritmos, herramientas, técnicas y metodologías de programación [97].

Finalmente, no podemos hablar de programación de computadores sin referirnos al término software como un programa o aplicativo compuesto de una serie de instrucciones que emplea datos para realizar tareas específicas mediante un sistema de cómputo para desempeñar hasta acciones dotas con patrones inteligentes a través del hardware provisto [98].

2.1.6 Estrategias utilizadas en la enseñanza-aprendizaje de la algoritmia

Una revisión sistemática de literatura [6] (Anexo 1) realizada en esta investigación, reportó las siguientes estrategias, tipificadas según su aplicabilidad en tres tipos:

colaborativas, centradas en el aprendizaje y centradas en la enseñanza, propuestas por los autores del presente estudio como lo muestra la Tabla 3.

Estrategias	Colaborativas	Centradas en el aprendizaje	Centradas en la enseñanza	Utilizan software
Estrategias de intervención	X		X	X
Pair programming	X	X		X
Evaluación de código por pares	X		X	X
Entorno virtual colaborativo inteligente	X		X	X
Ambientes colaborativos	X		X	X
MOOC		X		
Sistemas de visualización	X	X	X	X
Herramientas multimedia		X		X
Sistemas inteligentes de tutoría		X		X
Herramientas para aprendizaje visual		X		X
Juegos serios		X		X
Juegos educativos		X		X
Pseudolenguajes		X		X
Creación de videojuegos		X		X
Analogías			X	
Metáforas			X	
Robots			X	X
Aprendizaje Basado en Problemas (ABP)			X	
Realidad Aumentada		X		X
Realidad Mixta		X		X
Marcos de trabajo	X		X	X
Paradigma de programación y lenguaje de programación			X	X
Programación de procedimientos			X	X
Tutoría de compañeros		X	X	X
Lenguajes con una sintaxis simple			X	X
Curso de pre-programación			X	X
Herramientas de soporte			X	
Proyectos			X	
Entornos			X	
Desarrollo de Modelos mentales			X	
Enfoque del espiral			X	X
Máquinas de estados finitos			X	X
Clases magistrales y de laboratorio			X	X
Herramientas de calificación automática			X	X

Tabla 3. Tipificación de estrategias algorítmicas para un CS1
Fuente: elaboración propia

Estrategias de intervención. Silva *et al* [4] proponen combinar técnicas de aprendizaje colaborativo con programación de pares para el desarrollo de la lógica algorítmica en un CS1 con el propósito de fortalecer la formación de profesionales en construcción de software.

Pair programming. O programación colaborativa, se caracteriza por el trabajo de dos programadores de forma colaborativa en un computador, diseñando un mismo algoritmo y al mismo tiempo codificándolo y la probándolo [99] [100]. Pair programming es una propuesta pedagógica que se ha estudiado de forma amplia [101][102] y como resultado se ha comprobado que ha mejorado los procesos de aprendizaje en los cursos introductorios de programación, incrementando la confianza en sus integrantes haciendo del aprendizaje un momento divertido.

Evaluación de código por pares. Corresponde a una actividad colaborativa con el propósito de ofrecer retroalimentación en la fundamentación algorítmica a los estudiantes involucrados. La evaluación de código por pares puede convertirse en una estrategia que permite al estudiante desarrollar sus fortalezas y debilidades, plantear y cumplir objetivos en común, fortalecer la capacidad metacognitiva, su pensamiento crítico, y sus habilidades profesionales [103] [100].

Entorno virtual colaborativo inteligente. Es la integración de los Entornos Virtuales Colaborativos con la Inteligencia Artificial [100]. Por ende, es un sistema computacional creado para que un usuario interactúe con otros del mismo tipo o con partes del entorno con el fin de consolidar experiencias de aprendizaje algorítmico en condiciones espacio temporales distintas. El componente inteligente hace referencia a la interacción de bases de conocimientos mediadas con estrategias pedagógicas que dan cuenta a estados de indagación realizadas por el aprendiz [104].

Ambientes colaborativos. Los entornos colaborativos logran potenciar tanto el auto aprendizaje como fortalecer el razonamiento y el aprendizaje colaborativo [105]. Por esto, los ambientes colaborativos permiten que el aprendizaje de la lógica computacional se realice en un entorno con ciertos niveles de interactividad entre los estudiantes que beneficia directamente los procesos de adquisición de conocimiento [106].

MOOC. Corresponde a Cursos en línea masivos y abiertos que incluyen herramientas de discusión para que los estudiantes generen una dinámica activa en la discusión de temas puntuales [100]. La construcción de conocimiento para procesos algorítmicos se fundamenta en la discusión social generada como base del razonamiento pedagógico incorporado [107].

Sistemas de visualización. La visualización en procesos de software ayuda a representar de manera gráfica el contenido de un algoritmo o código fuente [108]. Hoy en día hay una amplia variedad de sistemas de visualización de algoritmos y programas que apoyan los procesos tanto de enseñanza como de aprendizaje de la algoritmia computacional [109]. Los sistemas de visualización permiten además, simbolizar ideas abstractas mediante elementos gráficos [110], para guiar al estudiante de una forma más simple tanto para la evaluación de su modelos como para la cimentación de su constructo en búsqueda del desarrollo de pensamiento algorítmico.

Herramientas multimedia. Mediante la integración de recursos como textos, imágenes, videos, etc. contribuyen con el proceso de aprendizaje del estudiante. Entre ellos se encentran: cursos dentro de un sistema de gestión de aprendizaje o LMS (Learning Management System) de programación y herramientas de software que mediante videos y screencast ayudan al estudiante en la búsqueda de ideas para resolver problemas de algoritmia computacional [111]; utilización de imágenes intuitivas que visualicen paso a paso la ejecución de cada instrucción de un código fuente [112]; la incorporación de un diario en el LMS como registro del proceso llevado a efecto durante el tiempo de aprendizaje [113], entre otros.

Sistemas inteligentes de tutoría [114]. Conformadas por herramientas de soporte a la escritura de código fuente en un programa computacional. Estas herramientas tienen mecanismos acordes a las capacidades de resolución de problemas con que cuente el estudiante. Entre ellas se encuentran: LispTutor, PROUST, MENO II, ELM-PE, ELM-ART, M -PLAT, CPP-Tutor, C++ STL y Prog-Tool.

Herramientas para aprendizaje visual [114]. Son herramientas que a través de representaciones gráficas de un algoritmo y/o del seguimiento en la ejecución de un código fuente, ayudan en el proceso de cimentación algorítmica computacional. Entre ellas se encuentran: Logo, Robot Karel20, JKarel Robot, Turingal, Scratch, Greenfoot, Alice, PLM, Robot Scribbler.

Juegos serios. Un número creciente de profesores/investigadores proponen la incorporación de juegos educativos (o juegos serios) en la enseñanza de la lógica algorítmica con el objetivo de reforzar la motivación instintiva a través de estudiantes desafiantes, despertando su curiosidad y dándoles una sensación de control e imaginación [17].

Juegos educativos. Una propuesta interesante para aliviar los problemas de diseño algorítmico es la incorporación de juegos educativos dentro de cursos de programación informática [17] como herramienta de enseñanza/aprendizaje.

Pseudolenguajes. Su utilización es importante en los CS1, estos tienen como ventaja que se puede configurar en idiomas locales y algunos se complementan con editores gráficos como apoyo a la abstracción algorítmica y pueden hasta generar código fuente en lenguajes de programación formal [66].

Creación de videojuegos. Consiste en creación de un entorno de juego partiendo desde cero y tomando como base la algoritmia computacional, contemplando adecuadamente metodologías que permitan integrar las buenas prácticas de programación ya conocidas [66]. La construcción de entornos de juego básicos permite una motivación mayor en el estudiante al momento de enfrentar un CS1 [25].

Analogías. Es una importante técnica utilizada por muchos profesores para la enseñanza de la algoritmia computacional, la cual toma la analogía para acercar al estudiante al concepto técnico y poder así buscar un significado más acorde a su experticia. Una analogía básicamente tiene un concepto fuente y el objetivo, donde el concepto que es familiar al estudiante es el llamado fuente y el resultante del proceso de abstracción se denomina objetivo [115]. Dunican [20] propone analogías para enseñar declaraciones a través de juguetes infantiles; para el manejo de listas con cajas, y el concepto de matriz mediante un casillero de correspondencia [5].

Metáforas. Son consideradas como una interesante herramienta didáctica y facilitan la enseñanza de un concepto abstracto [116]. En la ámbito universitario ha tomado interés la utilización de metáforas especialmente en los CS1 con el propósito de realizar un proceso de contextualización inicial para acercar al estudiante a un concepto abstracto [116] [117], lo cual está soportado por diversos estudios en los cuales por ejemplo se han utilizado metáforas para explicar conceptos como objetos, vectores, memoria dinámica, etc. [118].

Robots. Se han convertido en herramientas didácticas del aprendizaje constructivo en los cursos iniciales de programación ya facilitan el aprendizaje algorítmico y posibilitan la experimentación en tiempo real [110]. Uno de los mayores representantes en esta categoría son los kits de Robots Lego Mindstorms NXT mediante los cuales algunos profesores enseñan los fundamentos de programación [119]

Aprendizaje Basado en Problemas (ABP). Es un enfoque que ubica al estudiante como protagonista de en la adquisición de conocimiento realizando un aprendizaje por descubrimiento y construcción, el cual formula procesos algorítmicos desde sus propias actividades (la búsqueda de información, seleccionarla, su organización y la resolución de problemas) hasta los requeridos en cada una de ellas [120]. Existen varias experiencias reportadas en la utilización de este enfoque en la

enseñanza en las ciencias de la computacionales ya que la computación se basa en problemas y por la misma característica de actualización continua que conllevan los procesos tecnológicos [110].

Realidad Aumentada. La inclusión de la realidad aumentada como herramienta didáctica permite capturar de una manera fácil la atención del estudiante, permitiendo fortalecer los procesos algorítmicos mediante la combinación adecuada en la representación de los conceptos que debe enfrentar en un CS1 y así obtener resultados que favorecen el aprendizaje[121].

Realidad Mixta. Facilita el aprendizaje activo, permitiendo al estudiante interactuar con su entorno inmediato. Algunos profesores han combinado el aprendizaje de la algoritmia con procesos de realidad mixta mediante un entorno natural, buscando la adquisición de aprendizaje significativo [109].

Marcos de trabajo. Frittelli *et al.* [25] concluyen que la adecuada utilización de ejemplos de matemática y geometría como marcos de trabajo, introducen elementos favorables acordes con las vivencias de los estudiantes al momento de afrontar procesos algorítmicos en un CS1.

Paradigma de programación y lenguaje de programación. Robins *et al* [122] y Pears [123] resaltan la importancia de enseñar algoritmia en un CS1 combinando un paradigma con un lenguaje de programación de computadores.

Programación de procedimientos. Establece un estilo de aprendizaje sencillo e ideal para la formación algorítmica en un programador novel [124].

Tutoría de compañeros. Es una experiencia compartida entre una persona que ha tenido una apropiación específica de una temática (mentor de compañeros) y una persona que desconoce el tema (el compañero aprendiz) [100]. Este modelo colaborativo propicia un aprendizaje encaminado al beneficio muto con elementos innovadores y conlleva a superar problemas generalmente de comprensión ocasionados por la diferencia de conocimientos entre el profesor y el estudiante en el estudio de la algoritmia computacional [125].

Lenguajes con una sintaxis simple. Koulouri *et al* [126] con Mannila y De Raadt [127] recomiendan el iniciar con un lenguaje de programación como Python o Eiffel para la formación algorítmica, debido a que tienen sintaxis simples que facilitan el aprendizaje en las condiciones iniciales de un CS1.

Curso de pre-programación. Silva *et al* [4], Davies *et al.* [128] y Rizvi *et al* [129], consideran pertinente incluir un curso de pre-programación antes de comenzar con el curso de Programación de Computadores en el que únicamente se estudie el desarrollo de pensamiento algorítmico.

Herramientas de soporte. Silva *et al* [4] recomienda el uso de herramientas de soporte de programación con estudiantes noveles para así mejorara los procesos de compresión algorítmica de las temáticas estudiadas.

Proyectos. Buscan la manipulación de ciertos entornos simulados con el propósito de generar situaciones previstas y no planeadas para forzar al estudiante a resolverlas mediante algoritmos y así contribuir con su razonamiento lógico [66]

Entornos. Permite la construcción de entornos simulados desde cero mediante procesos algorítmicos, que incluyen modelamiento matemático que posteriormente el estudiante manipula y edita de acuerdo a los objetivos planteados [66].

Desarrollo de Modelos mentales. Baldwin y Kulijas [18] proponen que es posible el estudio de algoritmos en un CS1 mediante el desarrollo de métodos para construir o mejorar sus propios modelos mentales a través de interfaces de usuario, analogías y metáforas y la referenciación espacial.

Enfoque del espiral. Herbert [130] establece que la enseñanza más adecuada de la algoritmia computacional en los estudiantes, se hace a través de una explicación inicialmente ‘suave’ y luego se debe ir añadiendo mayor complejidad. Este proceso resulta ser extenso por lo que es necesario que exista una motivación constante durante todo el curso [5].

Máquinas de estados finitos. Para la enseñanza de la algoritmia computacional, Hartman, Nievergelt y Reichert [24], proponen las ‘Máquinas de Estados Finitos’ con el propósito de involucrar al estudiante en un contexto de juego. En este proyecto el estudiante puede aprender y simular el concepto de condicionales y ciclos. El objetivo de utilizar las máquinas de estado finito es la utilización de la gamificación en el proceso de aprendizaje algorítmico [5].

Clases magistrales y de laboratorio [110]**.** Son las más utilizadas en la mayoría de universidades e instituciones educativas para el desarrollo de pensamiento algorítmico, se caracterizan porque los temas estudiados en el aula de clase pueden reforzarse con herramientas didácticas en el tiempo independiente del estudiante, quedando el laboratorio como un lugar de encuentro dedicado a solución de ejercicios prácticos fundamentados en la teoría. En estos entornos, los estudiantes tienen un comportamiento generalmente pasivo, restringiendo en muchas ocasiones sus habilidades [131].

Herramientas de calificación automática [114]. Están dirigidas a la automatización de la calificación de ejercicios en el ámbito de la programación. Su propósito es contribuir a la realización de mayores cantidades de ejercicios por parte del

estudiante obteniendo retroalimentación rápida, a su vez para que el profesor dedique sus esfuerzos a la consolidación de la lógica algorítmica de programación. En este grupo se encuentran: Ceilidh, BOSS, CourseMarker, Web-CAT, BOSS2, SAC, Automata, eGrader, Pythia, CAP, AUTOLEP, Virtual ProgrammingLab (VPL), YAP3 + APAC, IT VBE y PETCHA.

2.1.7 Herramientas computacionales utilizadas para la enseñanza de la lógica algorítmica

Como resultado de la revisión sistemática de literatura realizada en esta investigación, en la Tabla 4 se presentan algunas herramientas de software utilizadas para la enseñanza de la lógica algorítmica en un CS1 clasificadas en cinco categorías.

Categoría	Herramientas
Visualización o simuladores de algoritmos	FLowchart INTerpreter o FLINT, Raptor, PSeInt, FreeDfd, Jeliot 3.
Herramientas de evaluación automática	TRY, PSGE, TRAKLA, CourseMaker, AutoLEP, Visual DaVinci
Juegos educativos centrados en la enseñanza de una unidad específica de aprendizaje	Catacumbas, Salvar a la Princesa Sera, EleMental: la recurrencia, Castillo de Wu, Robozzle, LightBot, TALENT, Gidget
Juegos educativos centrados en la enseñanza de unidades múltiples de aprendizaje	Robocode, M.U.P.P.E.T.S., Prog & Play, PlayLogo3D, Gidget, Scratch, Snap!, Train B&P, Entorno Cubik,
Ambientes colaborativos	EclipseGavab, Virtual Programming Lab (VPL), Ambiente Instruccional SABATO, COLLECE, COLLECE 2.0

Tabla 4. Categorías y herramientas para la enseñanza de la algoritmia en un CS1
Fuente: elaboración propia

2.1.7.1 Visualización o simuladores de algoritmos.

Son herramientas que permite hacer un seguimiento paso a paso de cada una de las instrucciones de un algoritmo de forma gráfica y/o mediante datos. En esta categoría se encuentran algunas de las siguientes herramientas:

FLowchart INTerpreter o FLINT [132]: Flowchart Interpreter o también FLINT, permite la representación de algoritmos mediante diagramas de flujo con sintaxis

mínimalista. FLINT facilita la retroalimentación continua de su código interpretado, y cuenta con herramientas de seguimiento propicias para programadores iniciales [4]

Raptor [133]. Es una herramienta tanto para aprendizaje como para la enseñanza de la lógica de programación, permitiéndole al estudiante modelar y codificar algoritmos de una forma gráfica, y a la vez también ejecutaros. Raptor no es un lenguaje de programación, sino un simulador de algoritmos provisto de una interfaz fácil de usar [4].

PSeInt. Herramienta de desarrollo de lógica computacional, diseñada especialmente para estudiantes novatos. Cuenta con una interfaz simple que mediante un pseudo-lenguaje, posibilita diseñar algoritmos de una forma sencilla con el propósito de que el estudiante afiance su estructura lógica procedimental [134].

FreeDfd. Es el sucesor del Smart DFD y permite modelar gráficamente un algoritmo o diagrama de flujo, a la vez es posible su ejecución con sus opciones de edición [134].

Jeliot 3. Es un visualizador para códigos Java, cuenta con una pantalla de seguimiento tanto de variables como de llamadas a métodos, la cual se refresca conforme avanza la secuencia de instrucciones, permitiéndole al estudiante determinar algorítmicamente y paso a paso la el comportamiento del código fuente [135][109].

2.1.7.2 Herramientas de evaluación automática.

Estas herramientas realizan procesos de verificación de instrucciones de forma estática y/o en ejecución tanto en editores de código como editores gráficos. En esta categoría se encuentran:

TRY [136]. Suministra al estudiante retroalimentación de forma instantánea, compara de forma algorítmica y paso a paso lo ejecutado versus lo esperado. Además, el estudiante puede realizar diversos intentos para alcanzar su respuesta bajo una restricción a cierto número de ensayos con el propósito de invitarlo a analizar sus procedimientos antes de volverlo a intentar nuevamente [2].

PSGE [137], surge como resultado de TRY, e incluye un módulo de conceptos de prueba de programas a través de el acrónimo SPRAE (Specification, Premeditation, Repeatability, Accountability, Economy) con el propósito de establecer un ciclo de vida de tareas de programación que pueden ser cuantificadas y calificadas de forma automática, además, dicho ciclo cuenta con tres etapas: especificación de tareas, un módulo para distribuir tareas y probar programas para extraer su algoritmo de acuerdo a un plan determinado [2].

TRAKLA [138] permite evaluar automáticamente algoritmos, y mediante su interfaz gráfica de usuario el estudiante construye sus propios diseños, limitándolo a un determinado número de intentos de ejecución establecidos en su configuración inicial con el fin de evitar que el estudiante abuse de la técnica de prueba y error [2].

CourseMaker [139]. Es auto compilado en Java y permite la configuración de seguimiento dinámico de interpretación de código en forma algorítmica y a la vez de pruebas estáticas de compilación. Además, cuenta con un módulo de registro del proceso de compilación en MySQL [2].

AutoLEP [140] realiza análisis estático en códigos e incluye el manejo de marcadores en puntos específicos que requieren análisis de comportamiento especial [2], el cual se hace paso a paso, lo que permite la extracción simbólica del algoritmo esencial.

Visual DaVinci. Es un entorno integrado de desarrollo (IDE) de programación para la representación y ejecución de algoritmos con el propósito de facilitar la enseñanza y el aprendizaje en CS1 y fue desarrollado por la Universidad Nacional de La Plata [3].

2.1.7.3 Juegos educativos centrados en la enseñanza de una unidad específica de aprendizaje [17].

Este enfoque toma como base los juegos ya construidos para la enseñanza de la algoritmia. A esta categoría pertenecen las siguientes herramientas de software:

Catacumbas Es un juego tridimensional multijugador que tiene como objetivo enseñarles a los estudiantes cómo declarar variables y usar algoritmos para declaraciones y bucles If simples y anidados [17]. El juego registra puntajes de experiencia para cada alumno y proporciona mensajes explicativos como un mecanismo de andamiaje [141][142].

Salvar a la Princesa Sera. Es un juego de dos dimensiones que permite a los estudiantes escalar a través de mensajes explicativos dirigidos al jugador [17]. Los estudiantes deben completar una serie de misiones para progresar en la trama del juego. Con este objetivo, completan las líneas de código que resultarán en un programa ejecutable. De esta forma, los estudiantes aprenden el algoritmo de ordenación rápida junto con bucles simples y anidados con el uso de un micro lenguaje [141][142]

EleMental: la recurrencia. Es un juego tridimensional que tiene como objetivo enseñar a los estudiantes cómo ejecutar algoritmos de recursión y de búsqueda transversal en profundidad, utilizando el lenguaje de programación C # [17]. Dos avatares llamados Ele y Cera ayudan a los estudiantes durante el juego de varias maneras. Por ejemplo, una vez que se escribe el algoritmo, Ele cruza el árbol binario según cómo se

implementa el código escrito, mientras que Cera explica exactamente qué está produciendo el código en un momento específico [143].

Castillo de Wu. Es un juego de rol bidimensional que tiene como objetivo enseñar a los estudiantes los algoritmos de ciclos y arreglos a través de actividades interactivas [17]. El juego permite la administración de matrices mediante el cambio de los parámetros dentro de los bucles y el movimiento de los personajes a través de la ejecución de bucles anidados [144].

Robozzle. Es un juego de acertijos en línea que proporciona una serie de instrucciones predefinidos, listos para usar y no muestra ningún código real [17]. Los usuarios pueden ejecutar sus instrucciones de acuerdo al algoritmo previo y ver cómo se mueve su héroe en todo el mundo y, por lo tanto, pueden detectar fácilmente los errores que han cometido y volver a reprogramar el algoritmo inicial [145].

LightBot. Es análogo a Robozzle y corresponde a un juego basado en acertijos y se puede jugar en línea. LightBot incluye un conjunto de comandos con los cuales el estudiante establece acciones algorítmicas, aunque no es un lenguaje de programación [146].

TALENT. Utiliza un micro lenguaje para la enseñanza algorítmica de enunciados y bucles. Talent utiliza por cada jugador un avatar en forma de arqueólogo mediante el cual el estudiante interactúa con el entorno virtual realizando actividades específicas y recolectando elementos para luego exponerlos en un museo [147].

Gidget [148]**.** Es una herramienta que consiste en programar algorítmicamente un robot, permitiendo la corrección de fallas para poder completar misiones establecidas previamente[66].

2.1.7.4 Juegos educativos centrados en la enseñanza de unidades múltiples de aprendizaje.

Al igual que la anterior categoría toman de base juegos ya desarrollados para realizar el proceso de enseñanza algorítmica. En esta categoría se suscriben:

Robocode. Es un entorno bidimensional que tiene como objetivo enseñar procesos algorítmicos usando el lenguaje Java. El juego se compone de un editor de programación, robots y una arena virtual, y los estudiantes deben programar la secuencia algorítmica de un robot que compita unos contra otros en la arena [17]. Durante su construcción, el robot hereda métodos básicos que luego pueden ser extendidos por los estudiantes de acuerdo con el comportamiento que quieren que tengan sus robots dentro de la arena [149].

M.U.P.P.E.T.S. Tiene por objeto preparar al estudiante para la asimilación de la concepción de la orientación a objetos a través de un entorno tridimensional colaborativo utilizando el lenguaje de programación Java. La metáfora de juego consiste en que el estudiante debe construir un robot el cual lucha en una arena virtual con otros robots mediante la escritura algorítmica de líneas de comando en un entorno de desarrollo integrado [150].

Prog & Play. Es un entorno de juego en el que los usuarios programan con algoritmos sus avatares, los cuales son héroes y quienes pueden conformar estrategias entre sí para poder subsistir la mayor cantidad de tiempo. Prog & Play, permite al estudiante elegir el idioma de codificación en Scratch, C, Ada, OCaml, y Compalgo [151].

PlayLogo3D. Es un juego de roles tridimensional que permite la interacción entre múltiples usuarios y tiene como objetivo enseñar la algoritmia fundamental y los conceptos básicos de programación informática estructurada [17]. El mundo virtual consiste en la nave espacial X-15 localizada en una constelación de la galaxia de Andrómeda, donde cada año se realiza un concurso entre pilotos-robots [152].

Gidget. Es un juego basado en la web donde los estudiantes pueden practicar algoritmia, utilizando un lenguaje de programación simplificado creado específicamente para el juego, con el fin de aprender a diseñar y analizar algoritmos básicos [17]. Un robot llamado Gidget tiene problemas con una parte de su software y, por lo tanto, no puede completar sus tareas y los estudiantes son llamados para ayudar a Gidget ya sea arreglando líneas de código incorrectas o completando el código que falta dentro de los programas [153]

Scratch. Brennan y Resnick [154][155] describen a Scratch como un entorno de programación algorítmico para la construcción de historietas, aplicaciones interactivas, que cuenta con una enorme comunidad en línea para compartir con otros usuarios.

Snap! [156][155] Es una extensión de Scratch y es un lenguaje de programación visual basado en algoritmos. Snap presenta opciones de configuración para abordar cursos iniciales de programación en cualquier nivel educativo.

Train B&P [157]**.** Mediante un sistema basado en ferrocarriles, apoya los procesos de aprendizaje algorítmico necesarios en un CS1 [66].

Entorno Cubik [158]**.** Posee un editor y traductor de algoritmos a código fuente, permitiendo la escritura de programas con enfoque estructurado o modular ya que se fundamenta en el paradigma imperativo [159].

2.1.7.5 Ambientes colaborativos

Son entornos de aprendizaje diseñados para la adecuada interacción entre estudiantes con el propósito de generar procesos de aprendizaje comunes**.** Los ambientes colaborativos reportados para un CS1 están:

EclipseGavab [160]. Es una versión de Eclipse personalizada pensada en el ejercicio profesoral. Esta versión combina aprendizaje colaborativo con el enfoque de Aprendizaje Basado en Proyectos, donde el estudiante convierte sus algoritmos en código fuente a través de lenguajes como Pascal, C y Java [106].

Virtual Programming Lab (VPL) [161]**.** Es un laboratorio para el aprendizaje de la algoritmia bajo la plataforma Moodle, que flexibiliza el desarrollo de software de manera virtual aprovechando los mismos recursos del gestor de contenidos. Esta herramienta fue diseñada por el Departamento de Informática y Sistemas, de la Universidad de Las Palmas de Gran Canaria [106].

Ambiente Instruccional SABATO [162]**.** Es una herramienta que combina el aprendizaje colaborativo apoyado por computador, conocido por sus siglas CSCL (Computer Support Collaborative Learning), y el Aprendizaje Basado en Problemas bajo las siglas ABP [106] para la escritura de algoritmos en un entorno colaborativo donde los estudiantes aprenden con sus propios pares.

COLLECE. Diseñando como soporte a la programación colaborativa en procesos tanto de desarrollo de software profesional como de enseñanza-aprendizaje en el aula de clase. Dispone de herramientas de edición remota para edición de algoritmos, código, control y asignación de tareas, entre otras [11].

COLLECE 2.0. Es una extensión de COLLECE, el cual integra el IDE de Eclipse en un entorno con varios plugin como recursos para programadores que mejora el entorno colaborativo de su antecesor [11] [163].

Como resultado del proceso de revisión literaria se presenta en la Tabla 5 un análisis más amplio de las herramientas algorítmicas utilizadas en un CS1.

Herramienta	Gráfico		Diagrama de flujo	Pseudo código	Genera código	Evaluación de estructuras de programación	Evaluación automática	Análisis (IO)	Trabajo Compartido
	2D	3D							
FLINT	X		X			X			
Raptor	X		X		X	X			
PSeInt				X		X			
FreeDfd	X		X		X	X			

Herramienta	Gráfico		Diagrama de flujo	Pseudo código	Genera código	Evaluación de estructuras de programación	Evaluación automática	Análisis (IO)	Trabajo Compartido
	2D	3D							
Jeliot 3	X					X			
Trakla	X		X			X	X		
PSGE	X					X	X	X	
TRY				X		X	X		
CourseMaker	X		X		X	X	X		
Visual DaVinci	X			X					
Catacumbas		X				X	X		
Salvar a la princesa Sera	X			X	X	X	X		
EleMental		X			X	X			
Castillo de Wu	X			X	X	X			
Robozzle	X			X		X			
LighBot	X			X		X			
Talent	X			X		X			
Gidget	X		X	X		X			
Robocode	X				X				
M.U.P.P.E.T.S.		X			X				X
Prog & Play	X				X				X
PlayLogo3D		X			X				
Scratch	X				X	X			X
Snap!	X				X	X			
Train B&P	X				X	X			
Entorno Cubik	X		X		X	X			
EclipseGavab	X			X	X	X			X
Virtual Programming Lab (VPL)	X			X		X			X
Collece	X			X	X	X			X

Tabla 5. Características de las herramientas algorítmicas para un CS1
Fuente: elaboración propia

2.1.8 Entornos educativos

La palabra entono de acuerdo al diccionario de la Real Lengua Española se define como el “ambiente” o el “conjunto de características que definen el lugar y la forma de ejecución de una aplicación” [164]. Asimismo, es el ámbito donde un individuo se desarrolla e involucra aspectos físicos, sociales, sanitarios, culturales, educativos, económicos, entre otros [165].

Desde el punto de vista educativo, la noción de entorno hace referencia a los espacios que permitan las interacciones educativas, es decir, aquellos espacios que faciliten la relación pedagógica [166]. A su vez, un entorno lo componen los profesores, estudiantes, el aula de clase, el espacio físico y hasta el entorno virtual dispuesto para el aprendizaje, junto con las metodologías, la comunidad escolar, la familia, los aspectos emocionales, la plataforma virtual, los materiales de trabajo, entre otras cosas [167]. Además, organizar y diseñar el entorno educativo requiere actividades como: organizar el espacio, planear el tiempo, los materiales, tener en cuentas las características de los estudiantes, y los aspectos pedagógicos y didácticos del profesor [168].

En la actualidad se habla de dos tipos de entornos: de enseñanza y de aprendizaje. Un entorno de aprendizaje hace referencia al espacio y los acuerdos establecidos para tal fin, donde la dimensión didáctica es la que formaliza como tal dicho entorno de aprendizaje, el cual se enmarca en cuatro dimensiones: social, física, técnica y didáctica [169]. Los entornos de aprendizaje son una nueva manera de aprender y tienen como actor principal al estudiante quien es el que regula su proceso de control y gestión del conocimiento [170].

Los entornos de aprendizaje también contemplan las diversas ubicaciones físicas, contextos y culturas en las que los estudiantes aprenden [171]. En el momento, existen varias adaptaciones del entorno de aprendizaje como los Entornos de aprendizaje (EA), Entornos Personales de Aprendizaje (PLE), Entornos Virtuales de Aprendizaje (EVA), Entornos de Aprendizaje Abiertos (EAA), entre otros.

Por otra parte, los entornos de enseñanza son dinamizados por el profesor y se centran en la en el aprendiz, el conocimiento y la evaluación, donde esta última busca facilitar oportunidades para la retroalimentación y la revisión, además, debe ser acorde con las metas educativas, los procesos y recursos. Al mismo tiempo, estos entornos buscan conseguir el equilibrio apropiado entre las actividades diseñadas para promover la comprensión y las diseñadas para promover la incorporación del conocimiento de forma práctica, además, si dichos entornos de enseñanza se centran en la experiencia de su comunidad, se favorecen el aprendizaje significativo de los estudiantes [172].

En este estudio es importante establecer la relación que puede existir entre un entorno educativo y un entorno de programación, el cual es un programa o conjunto de programas que, a través de varias tareas, permiten la construcción de una aplicación computacional. Además, estos entornos incorporan numerosas herramientas, utilidades, aplicaciones ya desarrolladas, ejemplos, tutoriales, etc., todas ellas encaminadas a facilitar y mejorar la construcción del software [173]. Por lo anterior, tanto los entornos

educativos como los entornos de programación tienen una relación compartida entre sí, tanto en su estructura como en los mismos propósitos.

2.2 Analogías

Las analogías son herramientas didácticas utilizadas muy frecuentemente en el sector educativo, y en la experiencia observada se puede determinar que algunos profesores las incluyen en sus currículos de forma planeada y otros las adoptan en la cotidianidad como parte de sus explicaciones en el aula de clase.

La analogía puede ser explicativa cuando plantea conceptos y principios nuevos en términos familiares y creativa cuando estimula la solución de un problema, la identificación de un problema nuevo y la generalización de los conocimientos [174].

Asimismo, una analogía establece semejanzas entre relaciones de la forma “A es a B lo que C es a D” y pueden ser exactas (“Buenos Aires es a la Argentina lo que Bogotá a Colombia”) o figurativas (“El presidente es al estado lo que el piloto al avión”) [175].

La palabra analogía fue inicialmente un concepto matemático que significaba proporción [176] y más tarde se consideró que no corresponde a una identidad de dos relaciones, sino que asegura una similitud de correlaciones [177], es decir, no supone igualdad simétrica, sino una relación usada con la finalidad de esclarecer, estructurar y evaluar lo desconocido a partir de lo que se conoce [178].

Las analogías son comparaciones entre nociones (conceptos, principios, leyes, fenómenos, etc.) que mantienen una cierta semejanza entre sí y su uso es muy frecuente en los contextos escolares en la comprensión de ideas complejas mediante situaciones que resultan más conocidas y familiares para el estudiante [178]. Asimismo, mediante las analogías se puede desarrollar la creatividad, la imaginación y las aptitudes y actitudes necesarias para el uso crítico de modelos científicos y para modelar la realidad por uno mismo [179][180][178]. Además, las analogías pueden facilitar la comprensión y visualización de conceptos abstractos que despierten y estimulan el interés por un tema nuevo y estimulan al profesor-investigador a tener en cuenta el conocimiento previo de los estudiantes [181].

Asimismo, “Trabajar con analogías implica, de alguna forma, una labor semejante al uso y construcción de modelos, por lo que requiere la búsqueda de conexiones entre objetos, atributos y relaciones entre ellos. Establece, por tanto, una cierta sistematicidad de pensamiento, un argumentar razones a favor y en contra de un postulado, y con ello también una forma diferente de ver el mundo, orientada desde criterios lógicos que van

más allá del pensamiento implícito de sentido común” [178].

Toda analogía presenta relaciones comunes entre el objeto y el análogo, pero también relaciones que difieren de un fenómeno a otro y en algunas ocasiones, los estudiantes tienden a interpretar la analogía en un sentido literal [178].

Las analogías son un tema de investigación relevante en el campo de la enseñanza y tiene un nuevo impulso a raíz de las concepciones del aprendizaje como proceso de construcción [182].

2.2.1 Metáforas

Por otro lado, el empleo de la metáfora es común en las corrientes psicológicas más actuales: constructivismo, terapia narrativa, terapias sistémicas, enfoques estratégicos y todas las nuevas tendencias que enfatizan la importancia del sujeto como creador y protagonista de su propia metáfora existencial [183]. Además, una metáfora es la aplicación de una palabra o de una expresión a un objeto o a un concepto, al cual no denota literalmente, con el fin de sugerir una comparación (con otro objeto o concepto) y facilitar su comprensión [184].

Una forma de construir metáforas se realiza cambiando una palabra de un contexto explícito por otra en un entorno diferente para realizar una comparación, por ejemplo, la expresión "El inicio de la vida", se puede sustituir 'inicio' por 'primavera' y así obtener la expresión metafórica: "La primavera de la vida" [185]. Otros ejemplos de metáforas son: las perlas de tu boca (perlas = dientes), …como ríos que brotan de tus ojos (ríos=lágrimas), estoy loco por ella (loco=enamorado), entre otras.

La razón por la que se propone el uso de metáforas es su utilidad probada como herramienta educativa cuando el dominio de enseñanza es abstracto [118], así, la “magia” de las metáforas no consiste en otra cosa que en proporcionar un momento inicial de motivación, facilitar el surgimiento de alternativas de acción o ilustrar con pinceladas iluminadoras un objetivo para mantenerlo en el punto de mira [183].

Sin embargo, las metáforas también deben tratarse con cuidado [118] ya que en usos previos de enseñanza basada en metáforas en Informática en niveles universitarios se ha advertido que no se debe llegar a “romper” la metáfora y es necesario usar cada metáfora en su ámbito de acción y cuidar adecuadamente su propósito [186].

En ocasiones se utiliza la metáfora como un sinónimo de analogía, pero existen varias diferencias entre estos dos términos, es así como la analogía realiza una comparación sin cambiar los conceptos en sí, mientras que la metáfora sustituye el término real con el que se compara [187][188]. Además, una analogía es una conexión

de dos situaciones con patrones de relaciones comunes entre ellas y tienen un carácter bidireccional y una metáfora es un tipo especial de analogía en el que la relación es unidireccional [189].

2.2.2 Modelo didáctico analógico.

El Modelo Didáctico Analógico (MDA) constituye una estrategia original de enseñanza que implica la construcción activa, por parte de los estudiantes, de los elementos del dominio base de la analogía [190], es decir, este modelo utiliza las analogías para explicar un tema específico a través del conocimiento que tiene un estudiante desde sus experiencias cotidianas [191]. Asimismo, Los modelos didácticos analógicos sirven como puente para la introducción provisional de un modelo aproximado evitando una visión excesivamente instrumentalista [192].

Además, el MDA es un modelo de analogías centrado en el profesor y en el estudiante, el cual a través de situaciones didácticas ayuda a los estudiantes a encontrar conceptos ya existentes en sus estructuras cognitivas, sobre los cuales construyen nuevos aprendizajes adecuados [193]. En este modelo didáctico, las analogías son comparaciones entre dos tópicos: uno nuevo (blanco) y otro conocido (tópico) y que se utilizan como herramientas del lenguaje para acercar a los estudiantes del conocimiento tradicional al conocimiento científico [194]. Construir un MDA requiere dominar con mucha profundidad el concepto a enseñar, debido a que es necesario identificar su esencia y las conexiones semánticas internas para llevar dicha situación, de una manera más cercana, a las vivencias del estudiante. Su aplicación requiere de 3 pasos: primero el MDA se presenta antes de abordar el tema específico, luego se enseña el concepto y termina en el proceso de metacognición que permite tomar conciencia del aprendizaje incorporado [195].

Al utilizar un MDA como mediador entre las experiencias previas y la rigurosidad conceptual de una temática, es posible evidenciar el cambio entre el modelo conceptual inicial y el final adquirido en el aprendizaje del estudiante al reconfigurar los conocimientos en su proceso de asimilación cognitiva [196].

Existe una fuerte relación entre los modelos mentales de los estudiantes que aprenden un concepto por primera vez y el modelo teórico objeto de dicho concepto, a través de la evolución de las ideas existentes frente a la nueva instrucción a asimilar, para ello, el MDA actúa como enlace en la primera aproximación acorde a la experiencias preconcebidas del estudiante y la formalidad rigurosa del concepto, cuando este presenta un cierto nivel de abstracción [192]

2.2.3 Analogías para la enseñanza de la programación de computadores

El aprendizaje de la programación mediante representaciones gráficas, con mayor o menor nivel de abstracción, es un campo en el que se está trabajando desde hace más de 25 años y cuyo propósito es reducir el nivel de abstracción que requiere la programación para facilitar su comprensión [197][109].

Las analogías y metáforas son herramientas didácticas utilizadas muy frecuentemente en el sector educativo, y en la experiencia observada se puede determinar que algunos profesores las incluyen en sus currículos de forma planeada y otros las adoptan en la cotidianidad como parte de sus explicaciones en el aula de clase.

La literatura científica reporta pocas experiencias en el uso de analogías para la enseñanza o aprendizaje de un CS1 con estudiantes universitarios, uno de los estudios es Collece 2.0 que es un entorno de aprendizaje de programación completo, basado en Eclipse, con capacidades de colaboración orientadas a la edición de proyectos en tiempo real, control de versiones, comunicación y otros elementos relacionados con la conciencia [198][109]. Este entorno se ha ampliado con técnicas de realidad mixta, aunque sin incluir técnicas de visualización mejorada de programas y algoritmos, que mediante el uso basada en carreteras y señales de tráfico permiten visualizar el flujo de ejecución de un programa de forma natural para el usuario, al encontrarse este familiarizado con ellas en su vida diaria [163].

A su vez, Sukamto y Megasari [199] desarrollan un modelo que convierte código fuente en imágenes de analogía mediante máquinas de estado para enseñanza de la programación. Asimismo, existen modelos basados en analogías para procesos de enseñanza en campos diferentes a un CS1, es así como Strunz y Louie [200] proponen un modelo de analogía entre el almacenamiento de energía y el tratamiento de datos en un computador para la enseñanza de conceptos de ingeniería eléctrica. Por su parte, Plappally [201] diseñó un modelo de analogías mediante mapas conceptuales para los cursos de un primer año de ingeniería mecánica. Stockdill et al., [202] implementan un modelo de analogías basadas en correspondencia para elegir representaciones de problemas matemáticos, entre otros.

2.2.4 Aprendizaje situado

Este tipo se aprendizaje tiene una relación estrecha con las analogías, Arias [203] afirma que la teoría del Aprendizaje Situado postula que existe una relación entre el aprendiz y el contexto, que se estructura sobre una base práctica, por ello, para que el

aprendizaje sea efectivo, el aprendiz debe estar activamente envuelto en un diseño de instrucción real. El aprendizaje Situado es también llamado aprendizaje anclado e cual es un proceso que representa una serie de cambios en las formas de comprensión y participación de los sujetos en una actividad conjunta y sobre un contexto pertinente, es decir es parte y producto de la actividad, el contexto y la cultura en que se desarrolla y utiliza [204], este tipo de aprendizaje está íntimamente conectado al constructivismo dialéctico, ya que en él se subraya la idea de que buena parte de lo que se aprende es específico de la situación en que se aprendió [205].

Capítulo 3

Entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico

En la Figura 3 se presenta el diagrama jerárquico de las temáticas abordadas en este capítulo.

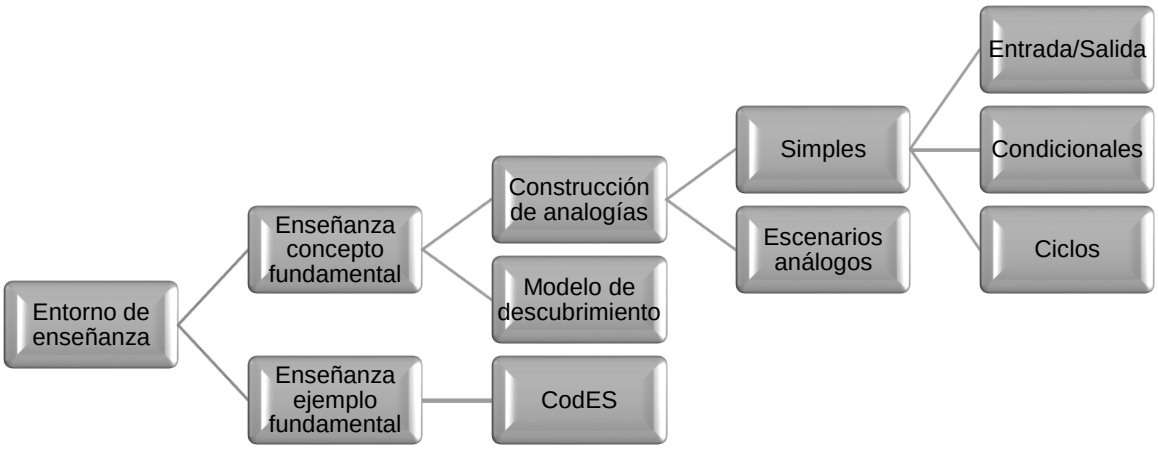


Figura 3. Diagrama jerárquico del capítulo 3
Fuente: elaboración propia

En este capítulo se propone un entorno de enseñanza para el desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios. El entorno de enseñanza aquí propuesto es un contexto planeado, dinamizado y dirigido por el profesor con la participación activa del estudiante, que incorpora procesos de planeación, estrategias didácticas, escenarios de trabajo (tiempo presencial e independiente del

estudiante), metodologías (para estudio de conceptos y ejemplos prácticos) y herramientas computacionales para facilitar el desarrollo de pensamiento algorítmico en un CS1.

Asimismo, este entorno de enseñanza articula los actores principales (profesor y estudiante) con estrategias didácticas de enseñanza (por parte del profesor) y de aprendizaje (centradas en el estudiante), utilizando como fundamento las vivencias o conocimientos prácticos del estudiante, los cuales son llevados al contexto de analogías para facilitar la abstracción de conceptos y a la vez es complementado con una herramienta de software para guiar al estudiante en la formación de su pensamiento algorítmico.

Este entorno de enseñanza incorpora una propuesta metodológica que combina un modelo didáctico analógico y una herramienta de visualización. Además, dicho entorno presenta unas habilidades que se sugiere debe poseer los estudiantes e incluye recomendaciones para los profesores encargados de orientar un CS1.

La propuesta metodológica para el entorno de enseñanza de desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios, basa su accionar fundamentalmente en el Modelo Didáctico Analógico (MDA) para la enseñanza de los conceptos fundamentales de la programación de computadores y una herramienta de visualización para el estudio de ejemplos fundamentales computacionales.

A su vez, el desarrollo de pensamiento algorítmico en potenciales programadores de computadores debe tomar como base los componentes estructurales del proceso de desarrollo de software para obtener en el estudiante resultados de aprendizaje evolutivos. Teniendo en cuenta que la apropiación de este tipo de pensamiento está limitado a generalmente pequeños periodos de tiempo (que habitualmente se expresan en créditos académicos), se requiere de una convergencia y sincronía adecuada tanto de los procesos de enseñanza como de aprendizaje, en la que tanto profesores como estudiantes tienen un rol fundamental. En la Figura 4 se presenta el entorno de enseñanza basado en analogías para desarrollo de pensamiento algorítmico de acuerdo a la definición propuesta.

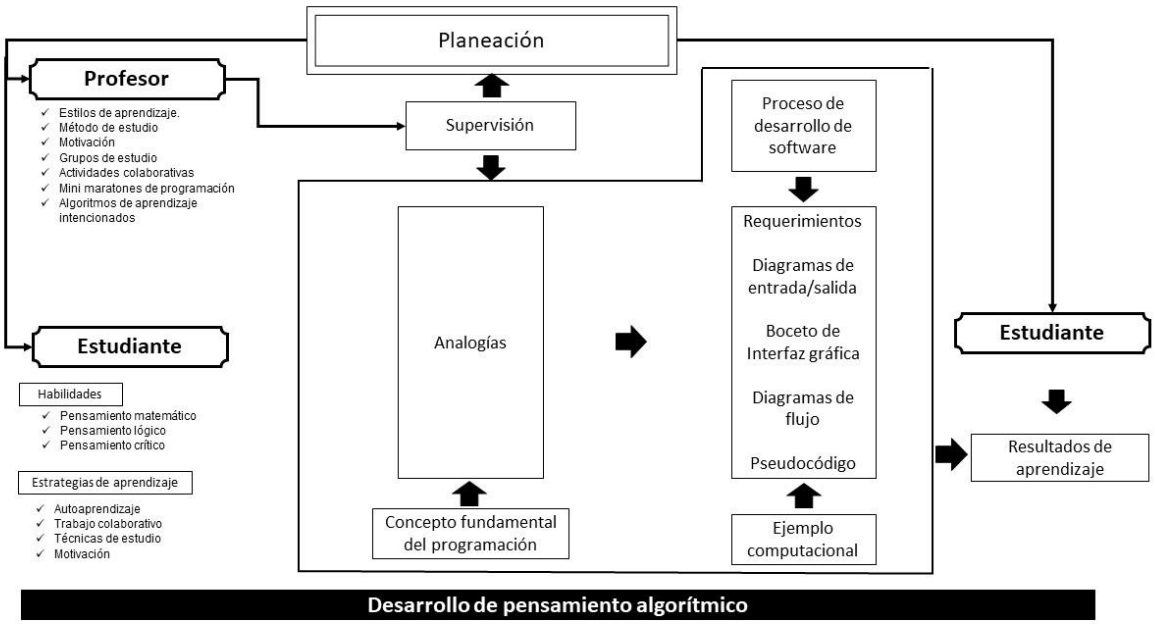


Figura 4. Entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico en un CS1
Fuente: elaboración propia

En el CS1, el desarrollo de pensamiento algorítmico debe ser cuidadosamente planeado por el profesor, debido a que los resultados de aprendizaje logrados por el estudiante serán la base para afrontar esquemas de pensamiento de mayor complejidad en los siguientes niveles de aprendizaje, como es el caso de desarrollo de la lógica de orden superior y la fundamentación del pensamiento complejo requeridos en la solución de problemas al estudiar y adaptarse tanto a nuevos paradigmas de programación como para el desarrollo algorítmico de soluciones que requieran el manejo de tipos de datos abstractos y su combinación con unidades de programa como rutinas, subrutinas, procedimientos, funciones, métodos, etc., para afrontar diversos tipologías algorítmicas existentes en la actualidad, como es el caso de los algoritmos de ordenamiento, de búsqueda, de vuelta atrás, de marcaje, de encantamiento, probabilísticos, heurísticos, de escalada, voraces, entre otros.

El desarrollo de pensamiento algorítmico es la clave fundamental en el aprendizaje de los diferentes tipos de paradigmas de programación, debido a que permite la adaptación del nuevo esquema mental en los procesos cognitivos de cada estudiante a tal punto de generar acercamientos a procesos metacognitivos cuando se alcanza altos niveles de

madurez como programador computacional.

En este estudio se recomienda que los estudiantes que afrontan un CS1 deben tener indicios de las siguientes habilidades para desarrollar un nivel inicial exitoso de pensamiento algorítmico: Pensamiento matemático estable, pensamiento lógico armonioso y pensamiento crítico en formación.

Dichas habilidades pueden ser examinadas inicialmente por el profesor antes de iniciar el CS1, con el propósito de identificar las capacidades del grupo de estudiantes frente al objetivo del curso, mediante la aplicación de test iniciales, que para esta investigación se realizaron mediante dos instrumentos: test de aptitud lógico matemático y test de aptitud de pensamiento crítico (Anexo 3).

A continuación, se describen las 3 habilidades:

- Pensamiento matemático estable. En la mayoría de casos, es la única habilidad formalmente desarrollada por los estudiantes durante la educación primaria y secundaria que contribuye con el desarrollo de pensamiento algorítmico.
- Pensamiento lógico armonioso. Es necesario para potenciar la capacidad de abstracción del estudiante que enfrentará conceptos propios de este campo.
- Pensamiento crítico en formación. Es importante que el estudiante tenga indicios de las siguientes destrezas: reflexión para asociar la nueva información, flexibilidad para buscar alternativas diferentes a las soluciones inicialmente planteadas, duda para seguir profundizando en la asimilación de conceptos y finalmente motivación y curiosidad que fortalecen el autoaprendizaje.

Además, es necesario que el estudiante también conozca algunas estrategias de aprendizaje que le permitirán apropiar de mejor manera la dinámica de un CS1, entre esas estrategias se encuentran:

- Autoaprendizaje. Es una característica primordial y su aplicación permite el desarrollo de estructuras mentales tempranas y apropiadas en la consolidación del pensamiento algorítmico, además, si esta habilidad no se desarrolla adecuadamente, se tiene el riego de que el estudiante demore su proceso de aprendizaje al obtener solo la información brindada por el profesor o tenga muchos vacíos conceptuales y procedimentales o en el peor de los casos sienta frustración al no poder alcanzar los resultados de aprendizaje desarrollados por sus pares de clase.
- Trabajo colaborativo. Esta estrategia es necesaria durante el proceso de formación en la consolidación del pensamiento algorítmico, ya que le permitirá al estudiante compartir el conocimiento desarrollado con sus demás

compañeros o caso contrario apropiar un conocimiento de una manera más cercana con el apoyo recibido por sus mismos compañeros a la hora de asociar un concepto abstracto mediante el lenguaje coloquial de sus mismos pares (algunas veces tiene mayor significancia la explicación brindada por un mismo compañero que por el mismo profesor).

- Técnicas de estudio. Es importante que el estudiante conozca al menos una técnica que le permita llevar a cabo procesos de apropiación de conocimiento y disposición para conocer otras.
- Motivación. Es una de las características fundamentales para el desarrollo de pensamiento algorítmico debido a que su aprendizaje puede tener un momento de inicio formal pero jamás tendrá un momento de finalización puesto que siempre habrá por descubrir algo nuevo y más aún con el vertiginoso avance de las ciencias computacionales.

La carencia de las anteriores habilidades y estrategias no inhabilitan al estudiante como candidato a desarrollador de software, sino que lo anima a equilibrar sus habilidades para que el proceso de aprendizaje sea más efectivo en el poco tiempo que dura un curso.

Por otro lado, así como los estudiantes deben tener unas habilidades y el conocimiento de ciertas estrategias de aprendizaje para desarrollar el pensamiento algorítmico, los profesores también deben tener en cuenta las siguientes recomendaciones al momento de enfrentar un CS1 para que el proceso de enseñanza genere las bases adecuadas en la consolidación de este pensamiento en el estudiante, entre esas recomendaciones se encuentran:

- Estilos de aprendizaje. Es necesario que el profesor de una forma ágil identifique el estilo de aprendizaje predominante en el grupo con el propósito de realizar orientaciones generales desde dicho estilo con acercamiento a los demás mediante herramientas de acompañamiento acordes con dichos estilos.
- Método de estudio. Es fundamental que el profesor recomiende a sus estudiantes los métodos de estudio pertinentes en cada momento del proceso de enseñanza/aprendizaje que permitan potenciar el desarrollo del pensamiento algorítmico.
- Motivación. El éxito de muchos cursos iniciales de programación depende del nivel de motivación inculcada por el profesor hacia sus estudiantes en la consolidación del pensamiento algorítmico como eje central del currículo del componente de programación presente en los programas académicos que forman desarrolladores de software.

- Grupos de estudio. Es recomendable en un primer curso de programación organizar grupos de estudio distribuidos estratégicamente, con el propósito de equiparar cada uno con estudiantes que demuestren mayor desarrollo de habilidades algorítmicas y así potenciar su aprendizaje mediante la réplica y al mismo tiempo apoyar a aquellos que lo requieran.
- Actividades colaborativas. Es importante que el profesor realice actividades tanto de aprendizaje como de evaluación en forma colaborativa al interior del aula de clase y fuera de ella (en el tiempo independiente del estudiante) con el propósito de obtener mejores resultados de aprendizaje en el grupo.
- Mini maratones de programación. No requieren de un lanzamiento como evento de participación masiva, sino que hacen referencia a la puesta en marcha de estrategias que le permitan a los estudiantes del curso competir entre sí de manera preferiblemente colaborativa en el desarrollo de actividades de aprendizaje, dicha estrategia debe permitir la participación total del grupo, de lo contrario únicamente premiaría a los que han desarrollado la competencia algorítmica y castigaría a los demás, lo cual sería un grave error.
- Supervisión. El acompañamiento continuo en el primer curso de programación por parte del profesor es muy importante, ya sea para resolver dudas o para asesorar al estudiante en la validación de los objetos de aprendizaje adecuados que permitan potenciar el desarrollo de su pensamiento algorítmico.
- Algoritmos de aprendizaje intencionados. Es necesario contar con un banco de ejercicios intencionados que incluyan niveles nulos, bajos, medios y altos de complejidad, los cuales deben ser presentados estratégicamente a los estudiantes con el propósito de reforzar su confianza y autoestima con los ejercicios de nivel nulo, para luego auto motivarlos con los ejercicios de baja complejidad y finalmente retarlos con los de mediana y alta complejidad.

Cabe aclarar que el pensamiento algorítmico es parte integral de las actividades desarrolladas por el ser humano debido a que desde el enfoque algorítmico todo lo que el hombre realiza en sus labores cotidianas se puede representar algorítmicamente, como el abrir una puerta, el cruzar la calle, el vestirse, el preparar la cena, el tomar un transporte, el preparar un examen, el calentar la comida, el preparar la fiesta de cumpleaños, el buscar una dirección, la navegación en internet para consultar una temática, etc., en fin, nuestro cerebro procesa algoritmos a cada instante, pero estos están tan automatizados que se realizan desde su memoria procedural sin pasar por los procesos de conciencia del cerebro.

En la Figura 5 se presenta el diagrama general de actividades para el Entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico, el cual contempla una configuración inicial (Anexo 2) desarrollada por el profesor para cada concepto fundamental de programación computacional de un CS1, que consiste en la planeación tanto de las analogías a utilizar en el artefacto construido como los ejemplos computacionales que se desarrollarán con CodES y los ejercicios complementarios requeridos para apropiar cada una de las temáticas.

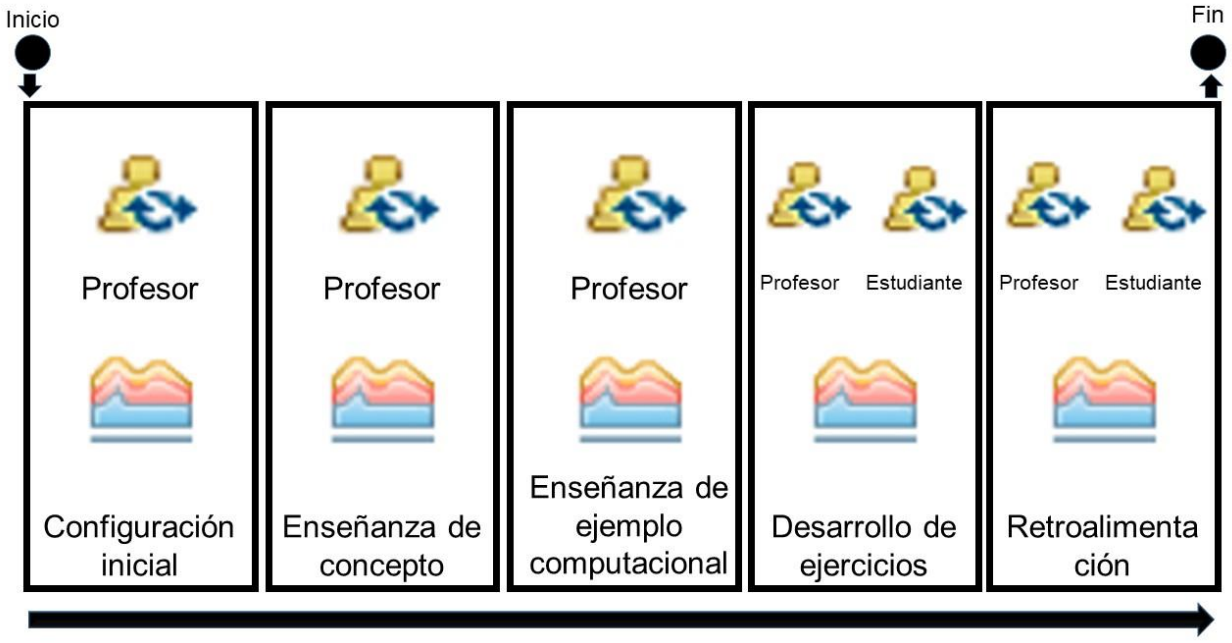


Figura 5. Diagrama de actividad general para el desarrollo de pensamiento algorítmico
Fuente: elaboración propia

Después de establecer la configuración inicial para cada concepto, el profesor utilizará la didáctica de enseñanza con analogías para dar a conocer el concepto fundamental de programación a los estudiantes de acuerdo a las sugerencias de construcción de analogías planteadas en este estudio.

Una vez presentado al estudiante el concepto fundamental de programación a través de una analogía, se desarrolla ejemplos computacionales con la estructura sintáctica y utilizando las palabras claves sugeridas en este estudio por cada temática a través de la utilización de la herramienta CodES.

Con la enseñanza del concepto fundamental de programación a través de analogías y la enseñanza del ejemplo fundamental computacional con CodES se desarrolló una serie de ejercicios que le permitieron al estudiante la práctica y apropiación de cada uno de los

conceptos presentados y cuyo proceso de validación se presenta en el capítulo 5.

Asimismo, en la Figura 6 se presenta el diagrama de actividad de detalle, en el que se especifica cada una de las actividades que se deben desarrollar tomando como base el diagrama de actividad de la Figura 5 para el profesor del entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico con estudiantes universitarios en un CS1.

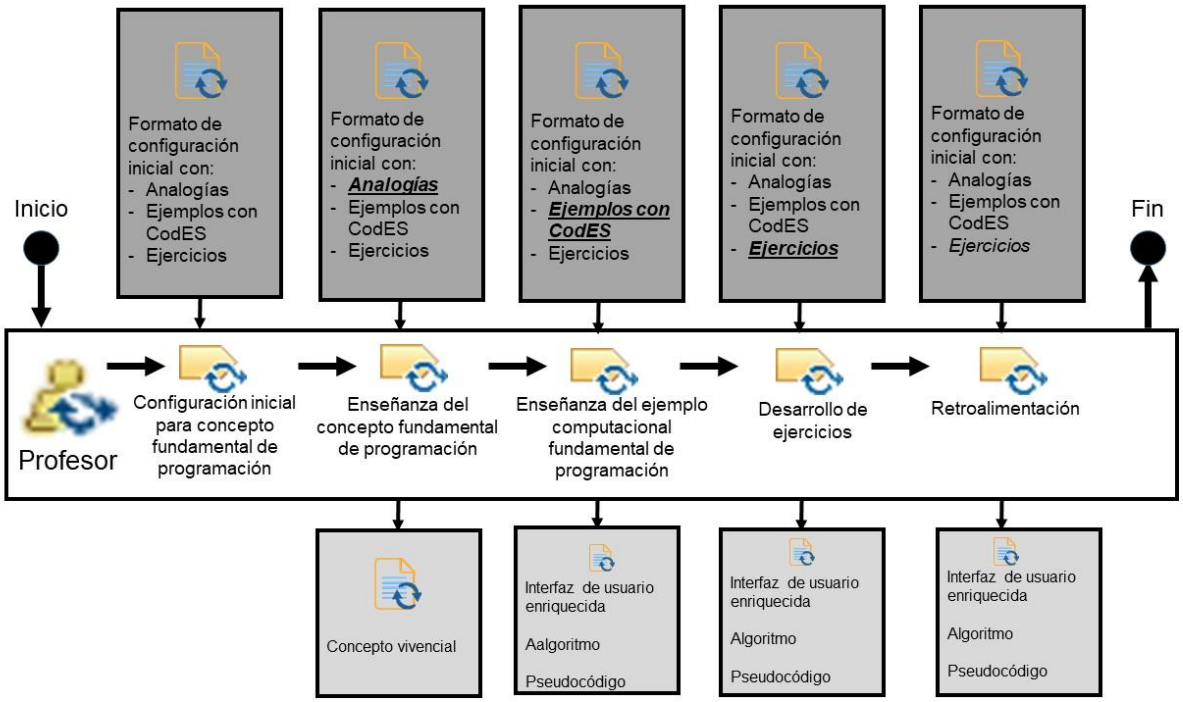


Figura 6. Diagrama de actividad de detalle para el desarrollo de pensamiento algorítmico
Fuente: elaboración propia

3.5 Propuesta metodológica para la enseñanza del concepto fundamental de programación

En este entorno se propone utilizar el modelo didáctico basado en analogías (MDA) el cual es ampliamente utilizado de manera formal o informal en la gran mayoría de los CS1, con el propósito de lograr en el estudiante procesos de abstracción de mayor nivel mediante análisis de experiencias situadas próximas a sus entornos vivenciales y que junto con las recomendaciones anteriores se pueda lograr un adecuado proceso de enseñanza en el CS1 y así poder obtener los mejores resultados de aprendizaje en el estudiante.

Es así como las analogías se proponen teniendo en cuenta el aprendizaje situado y el modelo didáctico analógico para acercar al estudiante novato a los conceptos

fundamentales de programación por parte del profesor para desarrollar pensamiento algorítmico como lo muestra la Figura 7.

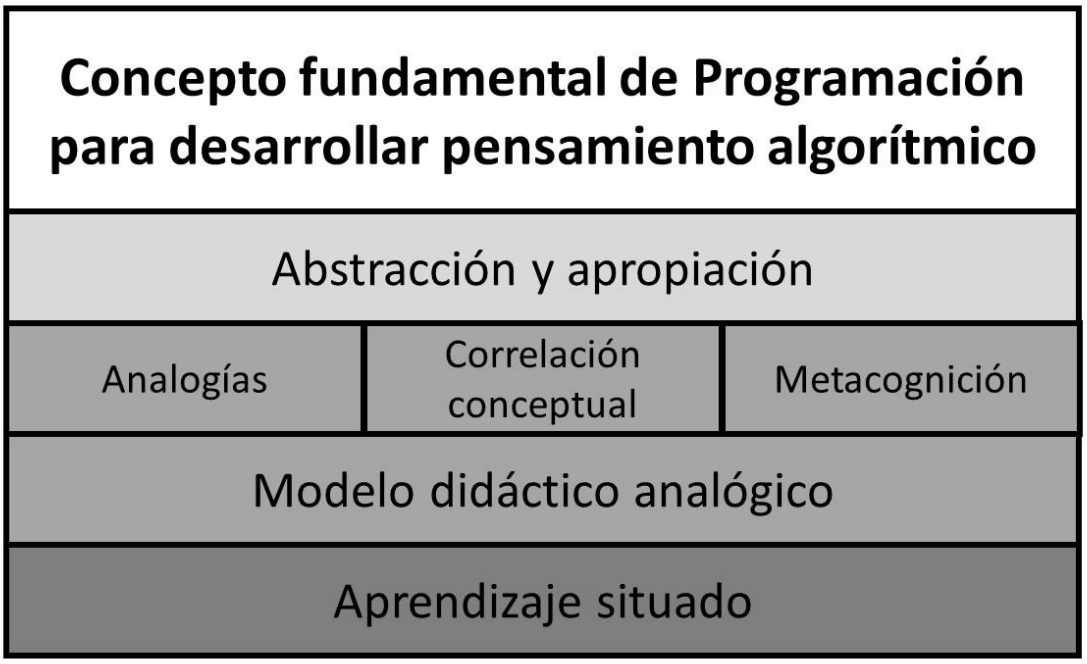


Figura 7. Modelo por capas para la enseñanza de los conceptos fundamentales de programación para el desarrollo de pensamiento algorítmico
Fuente: elaboración propia

El incorporar el aprendizaje situado en este modelo, permite que los ejemplos que se elijan para abordar el concepto fundamental de programación para desarrollar pensamiento algorítmico, deben ser del contexto cercano y de comprensión para el estudiante, incluso, incorporando expresiones o frases propias de cada región, pero que sean de pleno conocimiento por parte de este.

Por su parte, el Modelo Didáctico Analógico (MDA) aporta los elementos didácticos de enseñanza en torno a la utilización de la analogía como un elemento esencial, para acercar al estudiante a los diferentes conceptos requeridos en los fundamentos de programación de computadores para lograr de esta manera el desarrollo de pensamiento algorítmico en el estudiante, aprovechando sus estructuras cognitivas ya existentes sobre las cuales se formalizarán los nuevos aprendizajes. En este estudio se consideran los 3 elementos didácticos para la enseñanza con un MDA: 1) se presenta antes de abordar el tema específico, 2) luego se enseña el concepto y 3) se finaliza con la retroalimentación para verificar el aprendizaje incorporado

En la correlación conceptual se introduce el vocabulario técnico preciso a enseñar, el

cual se correlaciona con un listado de elementos presentados en el sistema análogo para que los estudiantes procesen la información, para así poder encontrar significado y comprensión por comparación con la información analógica vivencial presentada.

A su vez, la Metacognición es la ruta de estudio del proceso de enseñanza del profesor, que orienta la planificación, la ejecución y el control de acciones y operaciones mentales de los estudiantes que, en este caso, se han orientado conjuntamente con las etapas anteriormente mencionadas e, igualmente, se constituyen en una actividad para la regulación de la enseñanza del pensamiento algorítmico.

La analogía hace referencia a la estrategia de enseñanza basada en una situación real vivida por el estudiante en su contexto con el propósito de acercarlo de una forma didáctica al concepto fundamental estudiado. Es así como el elemento clave del modelo de capas de la Figura 9 es la analogía que utiliza el profesor para enseñar un concepto fundamental de programación. Para la construcción de una analogía se debe partir del concepto que se desea explicar, teniendo en cuenta los tres contextos análogos propuestos: Contexto de enseñanza situado en el aula de clase CESAC, Contexto de enseñanza situado y simbólico CESiS y Contexto de enseñanza situado y encubierto CESE, para lo cual es necesario utilizar situaciones vivenciales del contexto donde se realiza el aprendizaje, para que el estudiante identifique con facilidad dicha analogía como lo muestra la Figura 8.

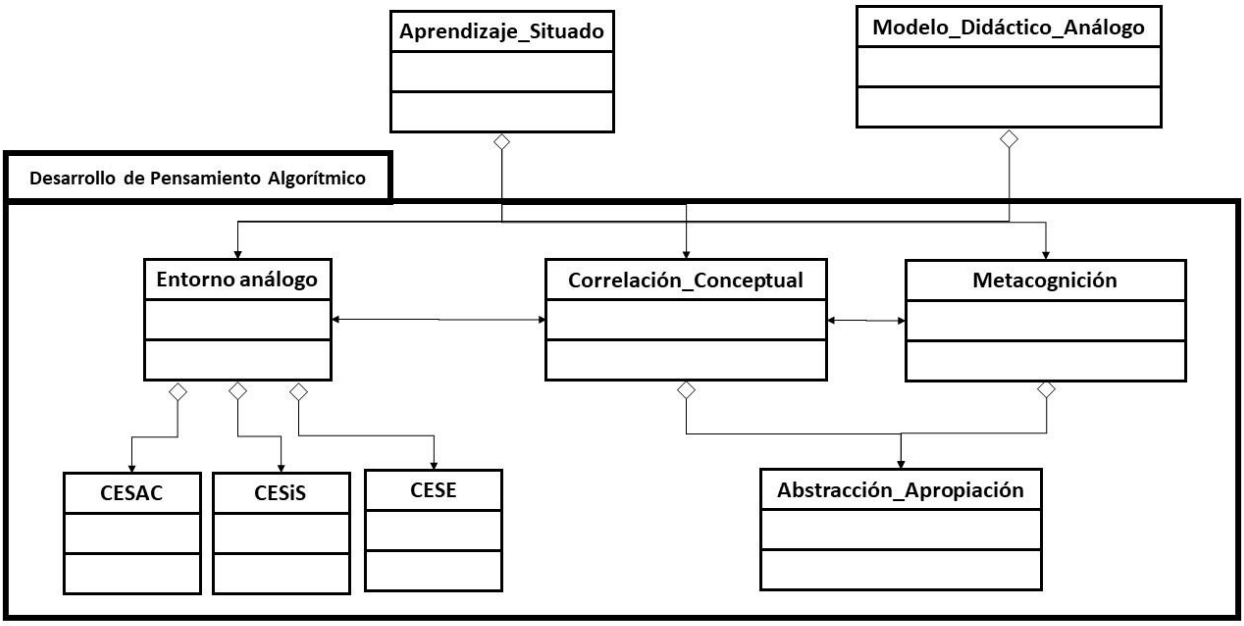


Figura 8. Modelo para la enseñanza de los conceptos fundamentales de programación
Fuente: elaboración propia

La enseñanza con analogías para este modelo tiene en cuenta los siguientes entornos:

1. Contexto de enseñanza situado en el aula de clase (CESAC): cuando la analogía se presenta con elementos físicos en el mismo salón de clase.

2. Contexto de enseñanza situado y simbólico (CESiS): cuando la analogía se le observa de forma indirecta, como en ambientes simulados, entornos inmersivos o digitales, videos, entre otros.

3. Contexto de enseñanza situado y encubierto (CESE): cuando la analogía se presenta cuando el estudiante que aprende lo hace imaginando la situación planteada.

La abstracción y apropiación de un concepto fundamental de programación de computadores a través de una analogía para desarrollar pensamiento algorítmico, se hace a través de un proceso de codificación y decodificación que parte de la situación vivencial que se ubica dentro del sistema real y mediante el proceso de metacognición se lleva al sistema formal como lo muestra la Figura 9.

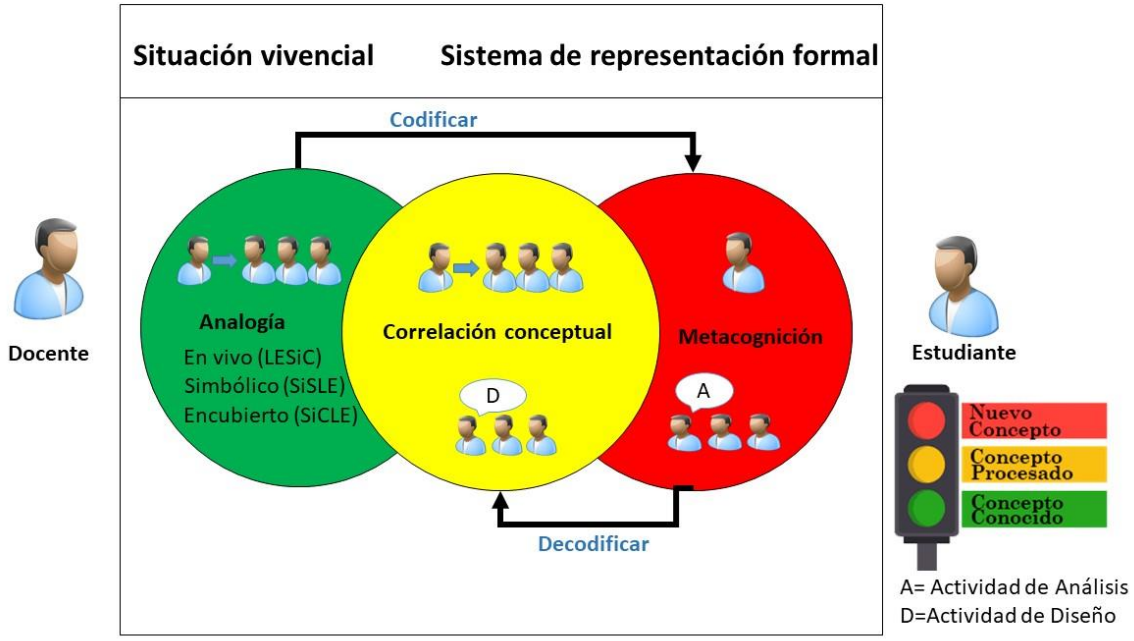


Figura 9. Proceso de codificación y decodificación de una analogía
Fuente: elaboración propia

La Figura 9 basa su accionar bajo la analogía de un semáforo, donde el color verde representa el conocimiento vivencial que tiene un estudiante y que forma parte de su dominio, el color amarillo es el proceso de correlación conceptual que se asemeja a una

zona de alerta que en este caso es donde el estudiante intentará relacionar el concepto nuevo con el presentado en la analogía y el color rojo es la zona compleja donde el concepto se formaliza y requiere de un “pare” cognitivo para ver si dicho concepto fue o no adquirido.

3.5.3 Construcción de analogías

En esta investigación se propone dos alternativas para la construcción de analogías:

- Analogía simple: consiste en utilizar analogías individuales para explicar los conceptos de los fundamentos de programación de computadores para el desarrollo de pensamiento algorítmico.
- Escenarios análogos: Consiste en utilizar un escenario para explicar todos los conceptos fundamentales de la programación de computadores para desarrollar pensamiento algorítmico. La Figura 10 representa el modelo propuesto.

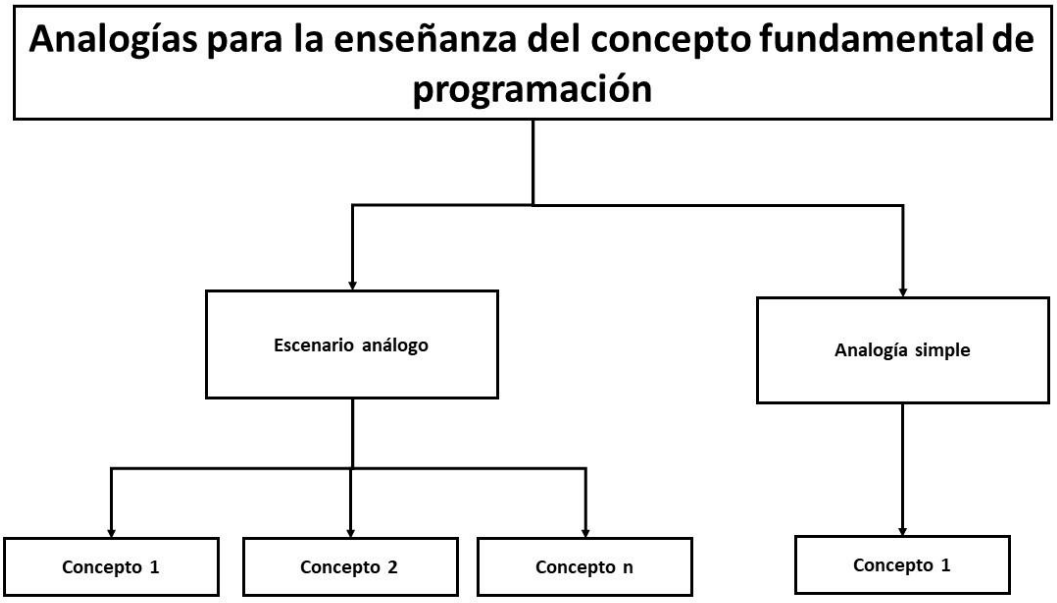


Figura 10. Tipos de analogías para la enseñanza del concepto fundamental de programación para el desarrollo de pensamiento algorítmico
Fuente: elaboración propia

3.1.2.1 Construcción de analogías simples

Para la construcción de una analogía simple se debe partir del concepto que se desea explicar, teniendo en cuenta los tres contyextos análogos propuestos (CESAC, CESiS y CESE) utilizando situaciones vivenciales del contexto donde se realiza el

aprendizaje para que el estudiante identifique con facilidad dicha analogía con el propósito de tener una apropiación más significante del concepto a aprender. En la Figura 11 se presenta el modelo de construcción general de una analogía simple.

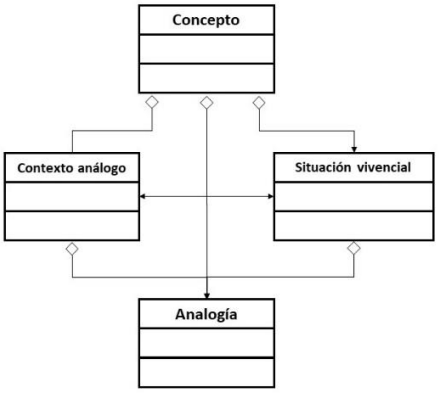


Figura 11. Modelo de analogía simple
Fuente: elaboración propia

3.1.2.1.1. Construcción de analogías simples para conceptos de Entrada/Salida

La construcción de una analogía simple para explicar conceptos relacionados con procesos de entrada/salida de acuerdo al modelo presentado en la Figura 11 requiere tanto la elección del entorno análogo a utilizar, como de la situación vivencial del estudiante como se muestra en la Figura 12.

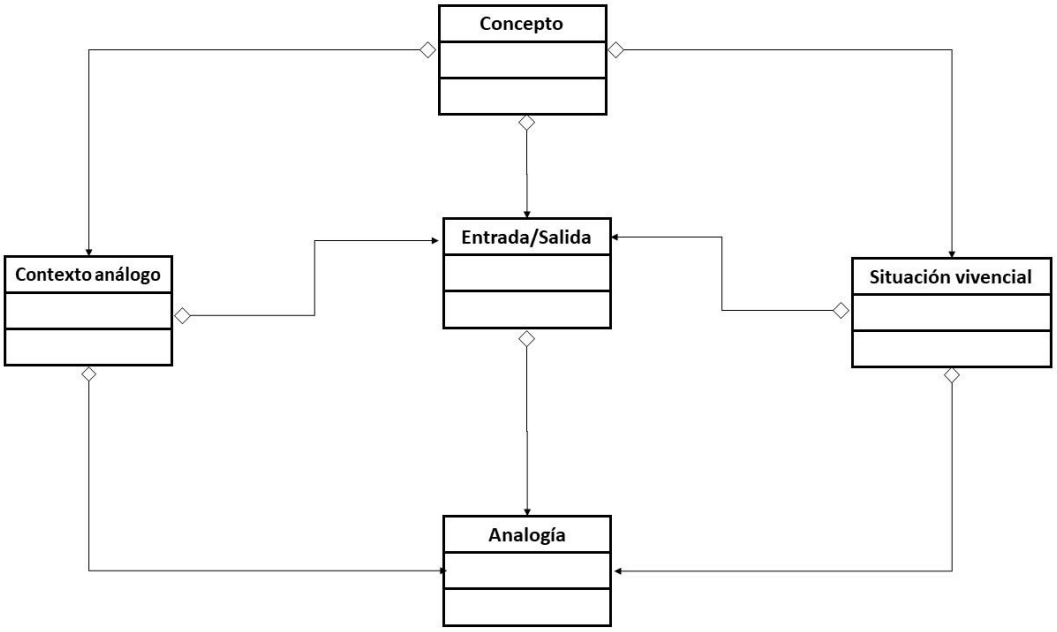


Figura 12. Modelo de analogía simple para Entrada/salida
Fuente: elaboración propia

Una vista más detallada del modelo de la Figura 12 es el que se presenta en la Figura 13, el cual detalla mediante el método “evaluar” los elementos necesarios que se deben considerar en la construcción de una analogía para la representación de un proceso de entrada/salida en la enseñanza de los fundamentos de programación para el desarrollo de pensamiento algorítmico.

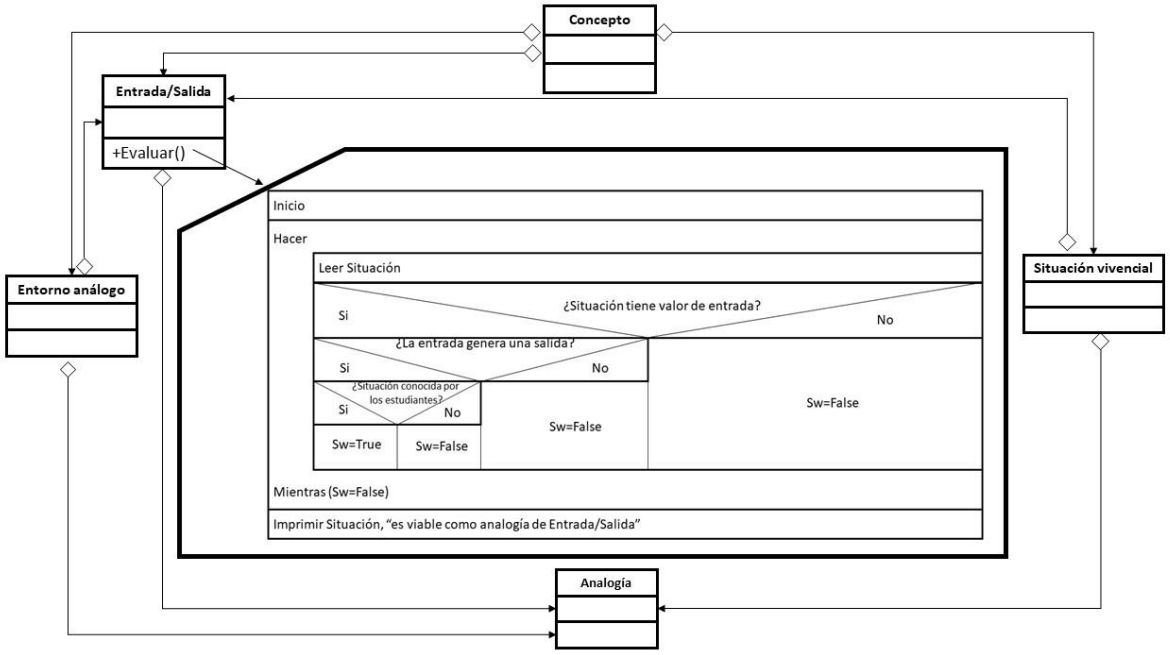


Figura 13. Método analogía simple para Entrada/salida
Fuente: elaboración propia

3.1.2.1.2 Construcción de analogías simples para condicionales

Al igual que la construcción de analogías simples para entrada/salida que de acuerdo al modelo presentado en la Figura 11 requiere un entorno análogo y una situación vivencial del estudiante, las analogías para condicionales también lo necesitan, como se muestra en la Figura 14 en la cual se considera la construcción tanto para condicionales simples como compuestos y anidados.

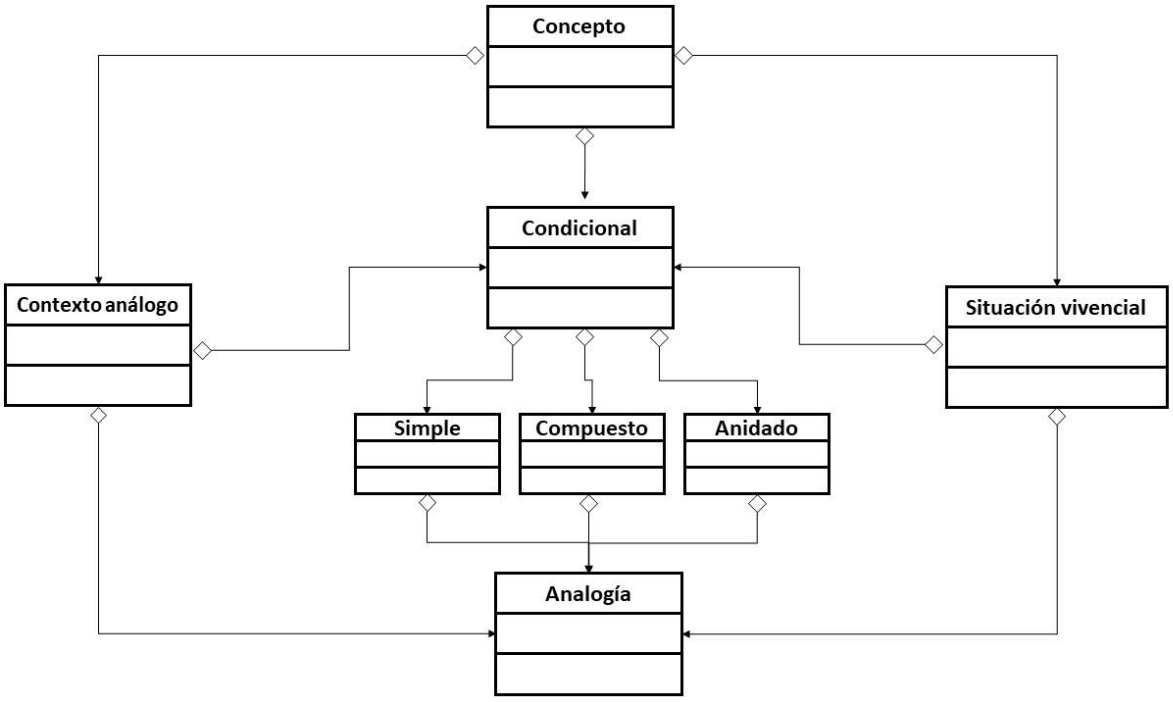


Figura 14. Modelo de analogía simple para condicionales
Fuente: elaboración propia

Asimismo, una vista más detallada del modelo de la Figura 14 se presenta en la Figura 15 el cual detalla mediante el método “evaluar” los elementos necesarios que se deben considerar en la construcción de una analogía para un condicional simple en la enseñanza de los fundamentos de programación para el desarrollo de pensamiento algorítmico.

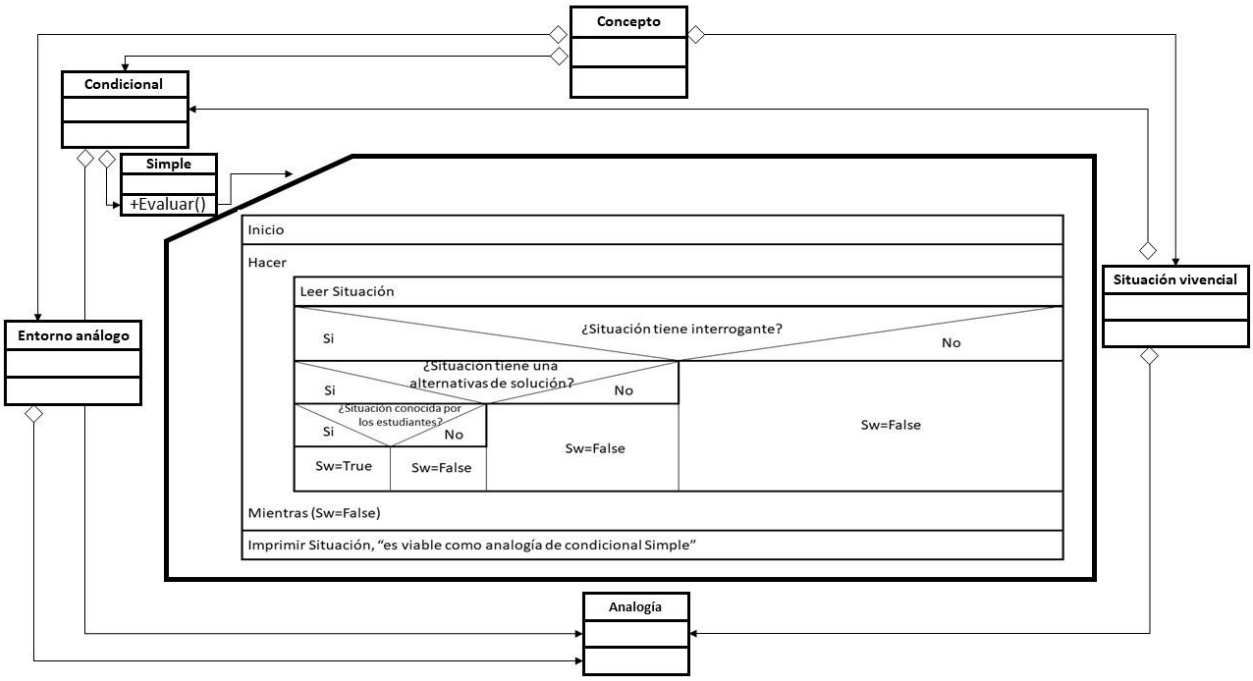


Figura 15. Método analogía simple para condicional simple
Fuente: elaboración propia

De igual manera, en las Figuras 16 y 17 se presenta la vista detallada los considerandos para la construcción de una analogía para un condicional compuesto y anidado.

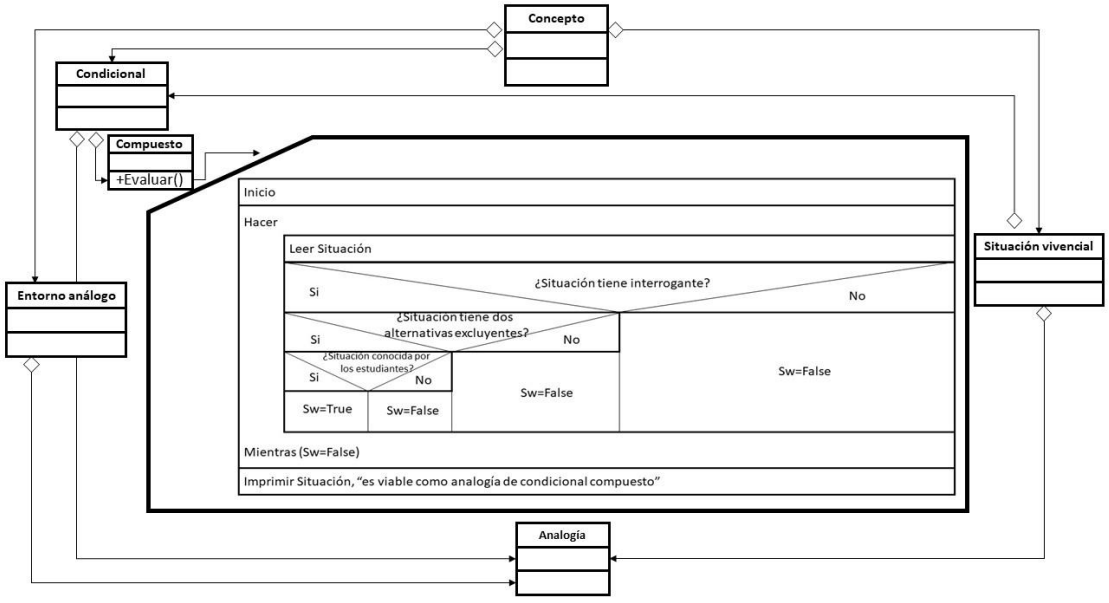


Figura 16. Método analogía simple para condicional compuesto
Fuente: elaboración propia

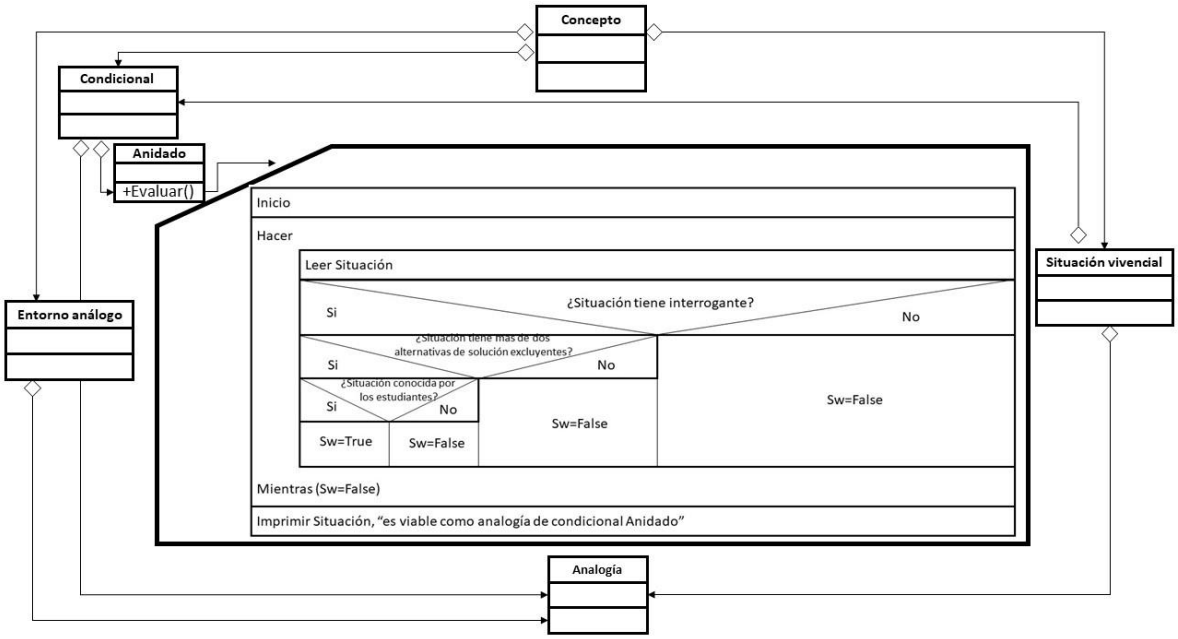


Figura 17. Método analogía simple para condicional anidado
Fuente: elaboración propia

3.1.2.1.3 Construcción de analogías simples para ciclos

De acuerdo al modelo presentado en la Figura 11, la construcción de una analogía simple para explicar conceptos relacionados con ciclos requiere de un entorno análogo y una situación vivencial, como se muestra en la Figura 18 en la cual se considera la construcción tanto de ciclo Para, Mientras y Hacer Mientras.

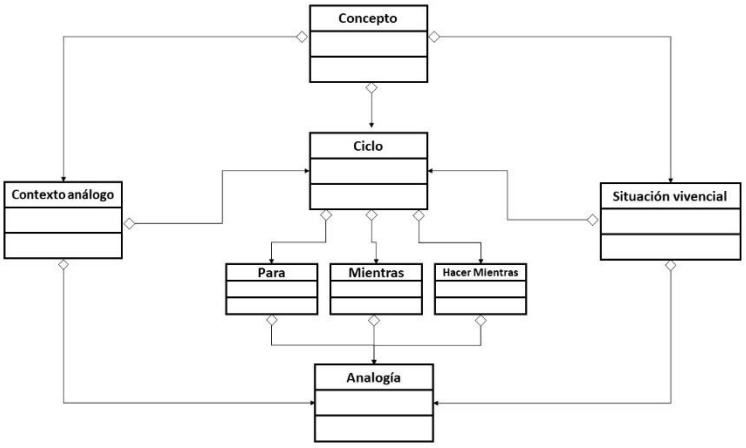


Figura 18. Modelo analogía simple para ciclos
Fuente: elaboración propia

Finalmente, en la Figura 19 se presenta la vista detallada de los considerandos para la construcción de una analogía simple para un ciclo (Para, Mientras y Hacer Mientras).

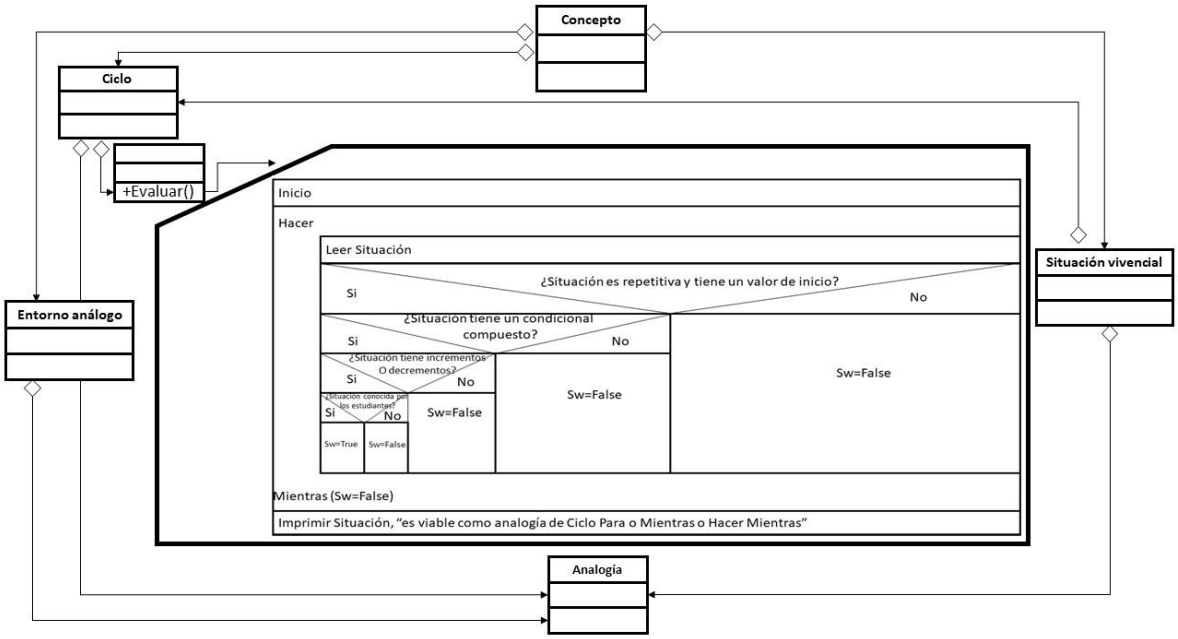


Figura 19. Método analogía simple en ciclos Para, Mientras y Hacer Mientras
Fuente: elaboración propia

3.1.2.2 Construcción de escenarios análogos

Para la construcción de un escenario análogo se debe analizar ambientes que contengan todos los anteriores conceptos y al igual que las analogías simples se debe tener en cuenta los tres entornos análogos propuestos (LESiC, SiSLE y SiCLE) combinándolos con situaciones vivenciales del contexto donde se realiza el aprendizaje. En la Figura 20 se presenta el modelo de construcción general de un escenario análogo.

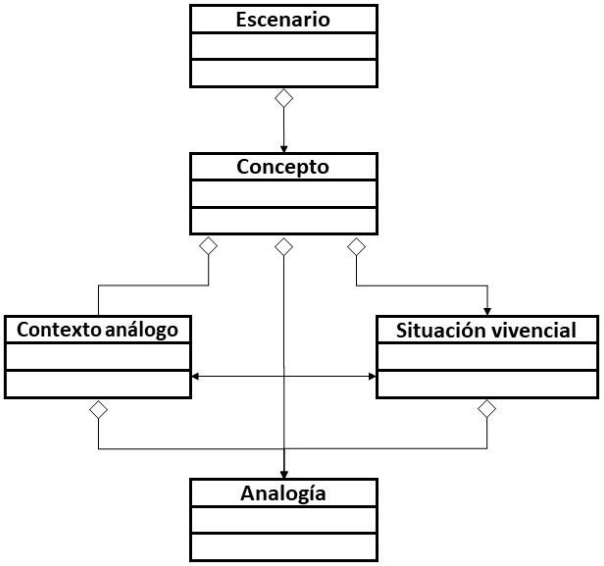


Figura 20. Modelo de escenario análogo
Fuente: elaboración propia

3.1.3 Modelo de descubrimiento para la construcción de analogías

En este numeral se presenta un modelo de descubrimiento de conocimiento que permitió la extracción de patrones, análisis lingüístico, analítica textual y datos enlazados al utilizar analogías para la enseñanza de los conceptos fundamentales de programación por los profesores en un CS1 en programas universitarios que forman constructores de software, con el objetivo de determinar los elementos que se deben incorporar en la construcción de una analogía para el desarrollo de pensamiento algorítmico. Para ello se planteó un modelo de descubrimiento basado en Machine Learning y Minería de Texto, utilizando técnicas de procesamiento de lenguaje natural (NLP) para modelado de espacio vectorial semántico, semántica distribucional y generación de datos sintéticos.

Para establecer una importante amplitud al proceso de descubrimiento se contempló la utilización de 13 métodos (9 de aprendizaje supervisado: J48, LMT, Jrip, Modlem, OneR, Part, SMO, InputMappedClassifier y IterativeClassifierOptimizer; 3 de no supervisado: Apriori, PredictiveApriori y K-Means y 1 semi supervisado: EM), con el propósito de tener una mayor cantidad de reglas, clasificaciones, asociaciones y segmentaciones que permitieron encontrar diferencias y similitudes en el banco de analogías analizadas, con el propósito de obtener patrones mejor diferenciados, que posteriormente fueron los que determinaron los elementos que debe contener una analogía para enseñar los conceptos fundamentales requeridos para el desarrollo de pensamiento algorítmico en un CS1.

Dicho modelo toma como entrada la estrategia didáctica de enseñanza con analogías para los conceptos fundamentales en un CS1 desarrolladas por los profesores en sus clases. Dicho modelo (ver Figura 21) consta de tres etapas: procesamiento inicial, descubrimiento y análisis.

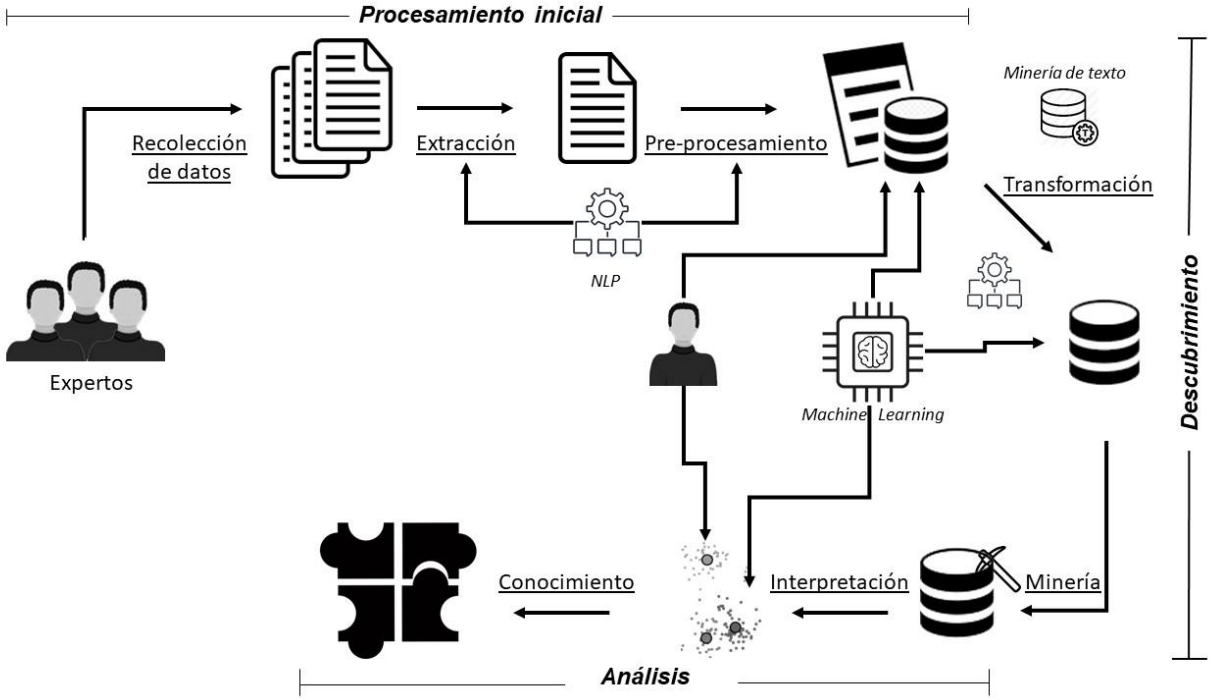


Figura 21. Modelo de descubrimiento de conocimiento
Fuente: elaboración propia

3.1.3.1 Etapa de procesamiento inicial

La etapa de procesamiento inicial consta de tres actividades: recolección de datos, extracción y pre procesamiento.

Para la recolección se utilizó como técnica la encuesta y se construyó un cuestionario semiestructurado cuyo objetivo fue caracterizar los procesos analógicos formales y no formales desarrollados en clase para un CS1 con profesores universitarios (Anexo 4).

Dicha encuesta estaba dividida en dos partes: la primera de ellas relacionada con datos generales de caracterización del profesor, universidad y del curso. En la segunda parte se indagaba sobre la experiencia de incorporación de analogías en cuatro conceptos fundamentales: entradas, salidas, condicionales y ciclos.

De este proceso participaron 15 universidades de tipo público y privado en 5 países de Centroamérica, Suramérica y Europa, con un total de 33 profesores expertos en CS1

en programas universitarios de computación que forman profesionales en construcción de software.

La actividad de extracción permitió realizar una caracterización de profesores y cursos, además, clasificó los ejemplos de analogías en cada uno de las unidades temáticas: Analogías para entradas, salidas, condicionales simples, condicionales compuestos, condicionales anidados, estructuras selectivas, ciclo para, ciclo mientras y ciclo hacer mientras. En esta actividad se contó con un total de 570 ejemplos de analogías utilizadas por los profesores en un CS1.

En la actividad de pre-procesamiento, se realizó la construcción de un Dataframe bajo una estructura de matriz de datos, al cual se realizó un proceso de limpieza mediante scripts en Python utilizando NLTK [206], dicha limpieza consistió en eliminar espacios en blanco al inicio, final, doble espacio entre palabras y eliminar caracteres especiales, convertir todas las palabras en minúsculas, eliminar acentos ortográficos, identificar palabras con plural y singular y corregir palabras mal escritas.

3.1.3.2. Etapa de descubrimiento

En la etapa de descubrimiento se realizaron dos grandes actividades: Transformación y Minería.

3.1.3.2.1 Transformación

Para la actividad de transformación se utilizó la técnica de minería de textos denominada Modelo de espacio vectorial semántico [207], mediante el enfoque de semántica distribucional [208] que analiza las palabras de forma individual, con actividades de Keyword Extraction, Summary Extraction, Tokens y Tagging Part of Speech (PoS).

Con las actividades anteriores se crearon los siguientes elementos gramaticales: Verbo principal y sustantivo. Además, con la participación de tres expertos se aplicó una técnica de clasificación para procesamiento de lenguaje natural bajo el modelo Intent Detection [209] que permitió extraer los siguientes campos semánticos: Acción, contexto y categoría.

Finalmente, en la transformación fue necesario recurrir al método de generación de datos sintéticos [210] para balancear el Dataset, para ello se utilizó NLTK de Python para generar sinónimos por cada variable contemplada y que permitieron balancear la cantidad de tuplas de los conceptos de entrada y salida frente a condicionales y ciclos.

3.1.3.2.2 Minería

Paso previo a iniciar la actividad de minería fue necesario particionar el Dataset en dos conjuntos de datos: Entrenamiento (para construir el modelo) y test (para evaluar su comportamiento). Para ello, se utilizó el muestreo estratificado y se tomó como parámetro la variable “Concepto” (entrada, salida, condicional y ciclo).Con estas consideraciones, se realizó una configuración inicial de 80% para Dataset de entrenamiento y 20% para test utilizando validación cruzada para obtener mejores resultados [211].

Por otro lado, en la actividad de minería se trazó como propósito la identificación de patrones que permitan al profesor utilizar analogías adecuadas para la enseñanza de conceptos fundamentales de programación de computadores, necesarios para el desarrollo de pensamiento algorítmico en un CS1. Para ello, y para obtener mejores resultados en la detección de patrones se aplicaron los tres tipos de aprendizaje: Supervisado, no supervisado y semi supervisado con sus correspondientes tareas [212].

En la actividad de minería se empleó el Dataset balanceado, y se inició con el tipo de aprendizaje supervisado mediante la tarea de clasificación [213], por ser una de las más utilizadas y que en este estudio tiene gran importancia debido al propósito de clasificar los datos en categorías que permitan descubrir patrones en las analogías utilizadas por los profesores para así obtener mejores resultados en la compleja tarea de enseñar los conceptos fundamentales en un CS1.

Por otro lado, el tipo de aprendizaje no supervisado se inició mediante la tarea de asociación [214] con el propósito el descubrir acciones frecuentes que se encontraban presentes en el banco de analogías utilizadas por los profesores en el CS1. Para esta tarea se utilizaron dos métodos: Apriori para buscar grupos de ítems frecuentes y PredictiveApriori para extraer las mejores reglas con parámetros de soporte y confianza.

Finalizando con la actividad de minería y con el propósito de obtener más información del conjunto de ejemplos recolectados como analogías empleadas en un CS1, se aplicó el tipo de aprendizaje semi supervisado mediante la técnica EM (Expectation Maximization) [215] que obtiene grupos mediante aproximación probabilística.

3.1.3.3. Etapa de análisis

Una vez finalizada la actividad de minería y para fortalecer el proceso de detección de patrones, se complementaron los hallazgos encontrados con tres técnicas adicionales:

Análisis lingüístico [216], analítica textual [217] y análisis experimental [218] por cada uno de los cuatro conceptos principales establecidos en este estudio.

Para el análisis lingüístico se aplicaron las técnicas de Lematización flexiva (que permite encontrar el lexema de las palabras analizadas) y Etiquetación morfosintáctica (asigna etiquetas a cada uno de los tokens evaluados de un texto, sin analizar las relaciones sintácticas) [219]; para la analítica textual se aplicaron cuatro técnicas de extracción: palabras clave, multi-palabras, entidades y analizador de sentimientos [220]. Finalmente, en el análisis experimental se utilizó la técnica de extracción de datos enlazados mediante tripletas [221].

Por ello, para la interpretación de resultados se utilizaron los hallazgos de las tres técnicas descritas anteriormente y los descubrimientos de la etapa de minería, soportados bajo el Modelo de espacio vectorial semántico con Tf-idf [207] con la librería OpenCv de Python, además, se utilizaron scripts para gráficas de frecuencias, diagramas de puntos, nubes de palabras, agrupamiento jerárquico con eliminación de términos dispersos y visualización de dendogramas por niveles (diagrama de árbol que clasifica los datos en categorías).

Asimismo, con WEKA se utilizaron los diferentes modelos de visualización de clúster, árboles y curvas de margen. Con estas herramientas fue posible complementar el proceso de extracción de patrones del banco de analogías utilizadas en la enseñanza de un CS1 por los 33 profesores vinculados en este estudio.

3.1.3.4 Elementos de una analogía para enseñar conceptos de los fundamentos de programación para el desarrollo de pensamiento algorítmico

El modelo de descubrimiento de conocimiento construido, generó como resultado la combinación de elementos morfosintácticos (integra la morfología y la sintaxis para construir oraciones con sentido y sin ambigüedad) con los hallazgos más importantes resultantes de la etapa de descubrimiento, obteniendo los elementos claves para la construcción de una analogía para la enseñanza de conceptos fundamentales de programación para desarrollar pensamiento algorítmico en un CS1, entre ellos se encuentran: la inclusión de palabras claves propias de la terminología de ejemplos computacionales, contextos cuantitativos, verbos en infinitivo, sustantivos comunes y el análisis sentimental de las oraciones.

Por lo anterior en la Figura 22 se presenta los elementos que se sugiere tenga una analogía para la enseñanza de dichos conceptos mediante la utilización de oraciones con sujeto elíptico o tácito las cuales no poseen en la estructura gramatical el sujeto explícito,

es decir, tienen la forma de instrucciones y se categoriza en el modo gramatical imperativo afirmativo.

Los hallazgos del aprendizaje supervisado, no supervisado, semi supervisado, análisis lingüístico, analítica textual y datos enlazados realizados, permitieron seleccionar los verbos, sustantivos y contextos adecuados para los conceptos de Entrada, Salida, Condicional y Ciclo presentados en este estudio con el propósito de acercar de mejor forma al estudiante a la abstracción de dichos conceptos.

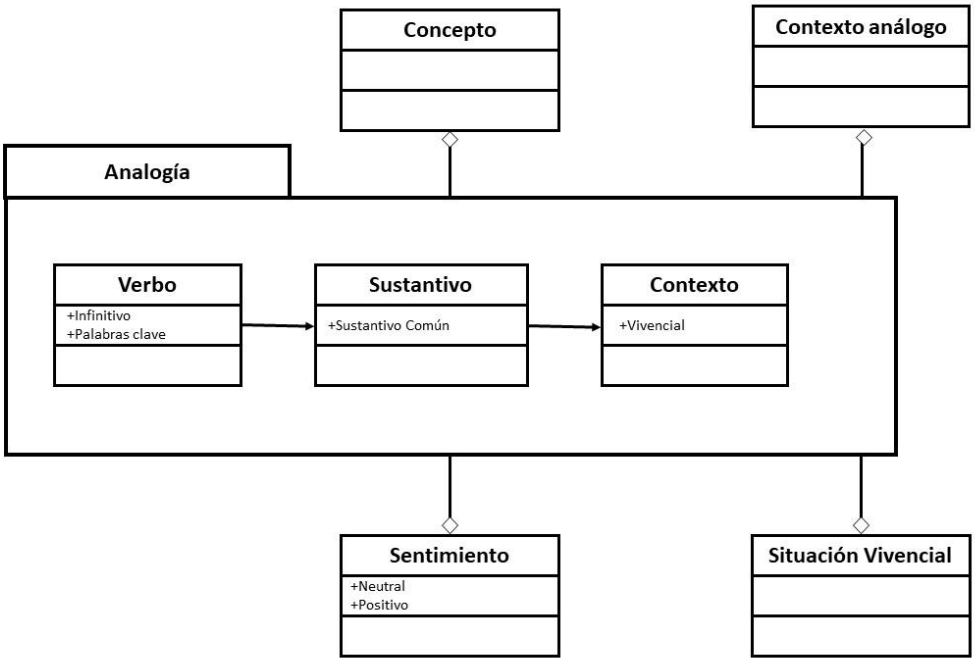


Figura 22. Elementos de una analogía para conceptos de programación
Fuente: elaboración propia

Por ejemplo, de acuerdo a los elementos sugeridos en la Figura 22, las siguientes analogías pueden ser utilizadas para enseñar los siguientes conceptos de los fundamentos de programación para el desarrollo de pensamiento algorítmico: digitar una clave en un cajero automático (entrada), obtener una factura por la compra de productos (salida), elegir un plato del menú del restaurante (condicional) y sumar los valores de cada producto de una compra (ciclo).

3.2. Propuesta metodológica para la enseñanza del ejemplo fundamental computacional para el desarrollo de pensamiento algorítmico

Este modelo establece que el desarrollo de pensamiento algorítmico en un CS1 debe apoyarse de los elementos claves utilizados en el proceso de desarrollo de software, ya que, desde este nivel, el estudiante debe comprender que la construcción de software obedece a una colección de actividades planificadas y bien diferenciadas.

Por lo anterior, desde el punto de vista de la computación, el pensamiento algorítmico es una habilidad que integra al pensamiento matemático, lógico, abstracto y creativo con elementos claves del proceso de desarrollo de software (requerimientos, análisis, diseño, codificación, pruebas y documentación) para modelar problemas y situaciones computacionalmente. En la Figura 23 se detalla el proceso de enseñanza en relación con las herramientas requeridas para apoyar el desarrollo de pensamiento algorítmico.

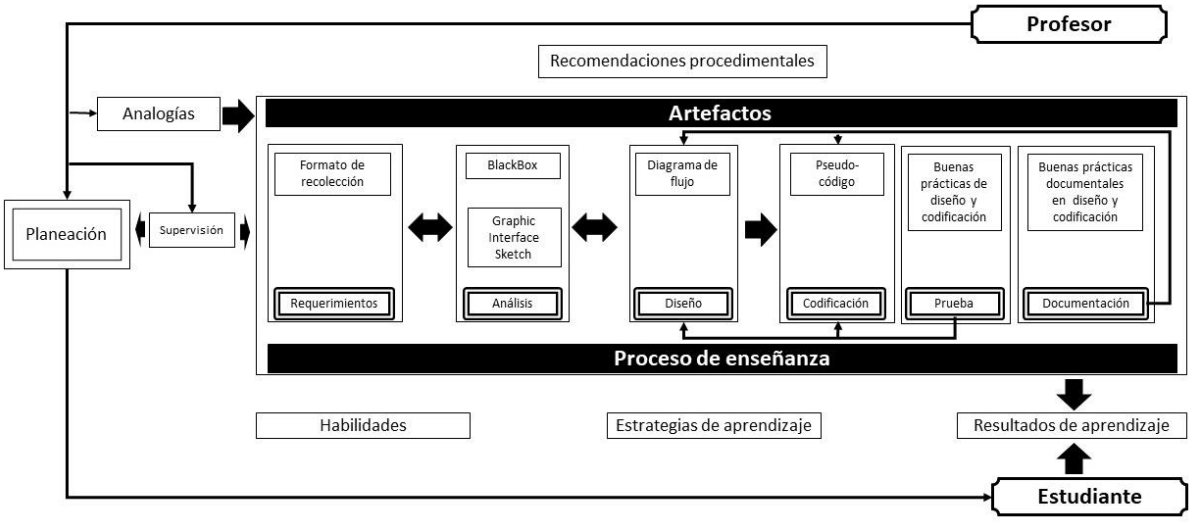


Figura 23. Modelo para la enseñanza de los ejemplos computacionales
Fuente: elaboración propia

Además, es necesario que el profesor realice la enseñanza en un CS1 a través del proceso de desarrollo de software donde el estudiante sea consciente de la importancia de cada fase, a continuación, se describe las actividades del proceso.

Requerimientos. Esta propuesta plantea un formulario básico que permite identificar los requerimientos del problema que deben estar de manera explícita en el enunciado del ejercicio planteado por su profesor y cuyo formato se encuentra en la Tabla 6.

Formato de recolección de requerimientos	
Nombre del sistema	
Requerimientos de entrada	- Capturar... - Capturar…
Requerimientos de salida	- Imprimir… - Imprimir…

Tabla 6. Formato para recolectar requerimientos
Fuente: elaboración propia

Análisis. Es necesario que el profesor le informe al estudiante que después de apropiar y realizar la fase de recolección de requerimientos (que para efectos de la presente propuesta se tomará a través del enunciado del problema) es necesario un proceso de análisis de la información, cuyo objetivo es determinar mediante la simulación de un entorno Cliente – Constructor de software donde el profesor asumirá el rol de cliente y el estudiante de constructor de software, la claridad que debe tener el constructor de software frente a las necesidades de un cliente para que el proyecto de software colme las expectativas que tiene su cliente, es decir, mediante esta fase el estudiante debe tener la certeza de que lo que piensa frente a la solución de la situación planteada (como constructor de software) es lo que realmente el usuario necesita.

Esta fase estará compuesta por dos actividades: construcción del diagrama Entrada/Salida (BlackBox) y la elaboración de la Interfaz gráfica de usuario enriquecida (Rich Graphical Interface).

BlackBox. Corresponde a un diagrama de Entrada/Salida permite el análisis de un sistema como si fuese una “caja negra” donde solo se tienen en cuenta las entradas y salidas de un sistema. El estudiante en este diagrama solo debe preocuparse por identificar y ubicar en el diagrama las entradas y salidas sin tener en cuenta el funcionamiento interno del sistema (ver Figura 24).

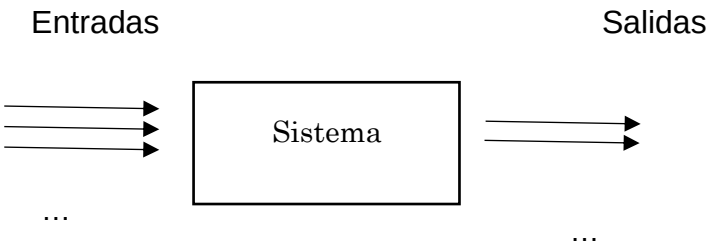


Figura 24. BlackBox
Fuente: elaboración propia

Para ilustrar al estudiante sobre el análisis con un BlackBox se recomienda que el primer ejemplo se realice con situaciones vivenciales y cada uno de ellos se asocie al concepto de “sistema” como lo son el uso de: cajero automático, calculadora, lavadora, horno microondas, entre otros, y verbalmente analizar sus posibles entradas y salidas.

En esta investigación, el BlackBox toma como entrada el formato de recolección de requerimientos.

Rich Graphic Interface (RGI). En esta fase se concibe una interfaz gráfica enriquecida de cómo se puede visualizar el diseño del software mediante una representación simbólica de ubicación de una serie de elementos utilizando el dispositivo de salida de un computador, es decir la pantalla.

El RGI permite representar los elementos analizados en un Diagrama de E/S. Además, el RGI utiliza una serie de símbolos que gráficamente sirve de guía tanto al programador para tener certeza del software que va a construir como al usuario de que su necesidad de software fue comprendida por el programador (ver Figura 25).

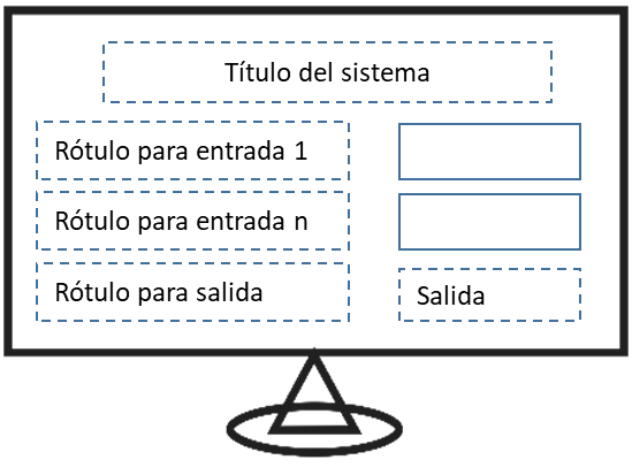


Figura 25. RGI

Fuente: elaboración propia

En la Figura 25, se utiliza como símbolo de entrada un rectángulo de línea continua tiene que ver con elementos reales de construcción de Interfaz gráfica de usuario (GUI) incorporados en los lenguajes de programación actuales haciendo una similitud con los campos de texto que permiten la captura de datos por teclado (como por ejemplo los JTextField de Java o los TextBox de .Net). Así mismo, el símbolo de salida utilizado corresponde a un rectángulo de línea punteada que cumple las veces de una etiqueta (por ejemplo, un JLabel en Java o un Label en .Net).

En esta investigación, el RGI toma como entrada el BlackBox.

Diseño. En esta fase el profesor le informará al estudiante quien ya tiene claro la fase de recolección de requerimientos y la fase de análisis, que es importante construir un modelo que cumpla con los requerimientos exigidos y con los elementos realizados en la fase de análisis.

De esta manera en la fase de diseño se construirá un modelo basado en algoritmos utilizando un diagrama de flujo que tome como base la información existente en la fase de análisis y que presente una solución que puede ser tratada como un proceso algorítmico y que puede ser llevada a una representación más abstracta basada en el modelado Orientado a Objetos.

En esta investigación, el algoritmo toma como entrada el RGI.

Codificación. El profesor realizará la codificación del diagrama de flujo realizado utilizando únicamente el pseudocódigo, con el propósito de generar un primer código simple y acorde al diseño planteado (el hacerlo en un lenguaje de programación formal puede conllevar al estudiante a desviar su atención en la cimentación de su lógica).

En esta investigación, el pseudocódigo toma como entrada un algoritmo.

Prueba. Es necesario que el profesor ilustre al estudiante mediante las buenas prácticas de codificación (así sea en pseudocódigo) y realice pruebas constantes tanto en el diseño del diagrama de flujo como en su escritura.

Documentación. Se recomienda que el estudiante aplique buenas prácticas de documentación que deben ir desde el mismo diseño del algoritmo en el diagrama de flujo como en la misma codificación en pseudocódigo, utilizando los comentarios de una sola línea o multi-línea e iniciando con la escritura del enunciado del problema en el mismo script de codificación.

En la Figura 26 se presenta el detalle de actividad tanto para el profesor como para el estudiante para la enseñanza de la programación de computadores en un primer curso y teniendo como base el proceso de desarrollo de software.

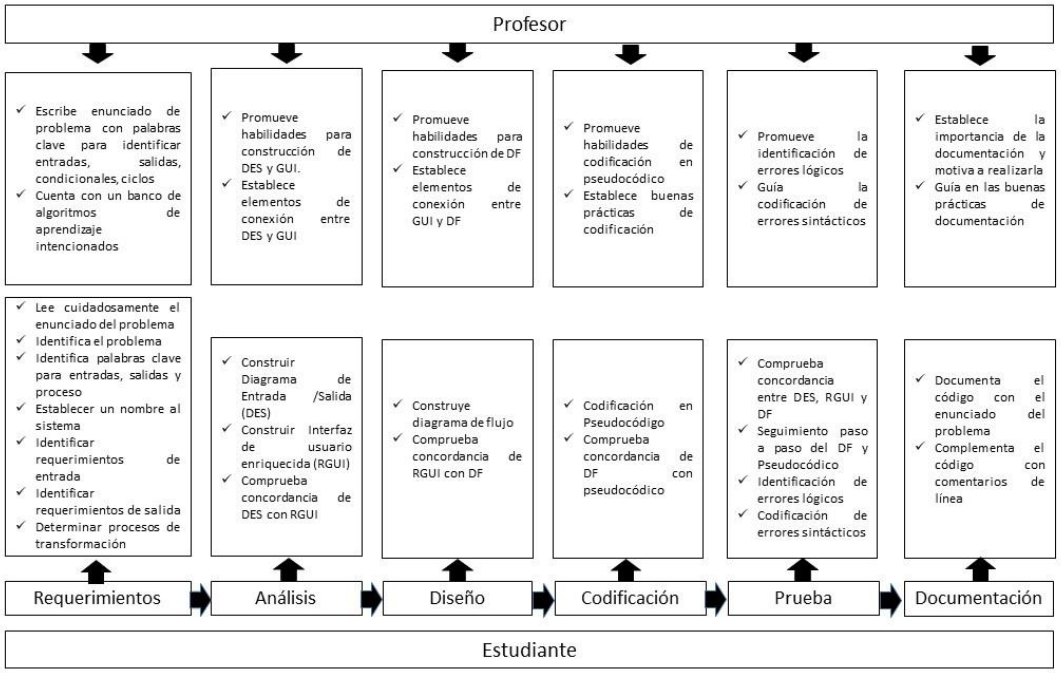


Figura 26. Detalle de actividades en proceso de desarrollo de software
Fuente: elaboración propia

3.2.1 CodES: Herramienta de visualización

Basados en la propuesta metodológica para la enseñanza del ejemplo fundamental computacional en esta investigación se construyó CodES (CODificación con Entradas y Salidas). Esta es una herramienta de visualización que basa su accionar en el artefacto más simple de análisis computacional que es el diagrama de entrada/salida que toma como insumo el formato de recolección de requerimientos, con el propósito de generar procesos de abstracción para el diseño y escritura de algoritmos computacionales, permitiendo que el estudiante centre su atención en comprender el problema a solucionar mediante sus elementos esenciales y a la vez intuir desde un inicio la interfaz computacional a construir con sus diagramas de diseño y codificación.

CodES basa su funcionamiento en el diagrama de entrada salida y propone una estrategia didáctica de enseñanza basada en palabras claves de identificación para que el estudiante reconozca si el enunciado del problema a solucionar es de transformación

de salidas o tiene estructuras condicionales o cíclicas. Dichas palabras claves son los verbos de mayor frecuencia que fueron resultado del estudio realizado en el modelo de descubrimiento para las analogías utilizadas en un CS1.

El entorno de trabajo de CodES consta principalmente de los siguientes componentes (Figura 27):

- **Toolbar**. Provisto de cinco elementos básicos: entrada, proceso, salida, rótulo y continuidad; mediante los cuales se realiza todo el proceso de desarrollo inicial de pensamiento algorítmico.
- **BlackBox**. Es el principal componente de CodES y es donde el usuario construye el diagrama de entrada salida con los 5 elementos básicos del Toolbar
- **Rich Graphical Interface (RGI)**. Permite visualizar cómo será la interfaz inicial computacional de los elementos ubicados en el BlackBox. Además, esta vista le presenta al estudiante cómo distribuir básicamente dichos elementos en una interfaz de usuario y los corresponde con el BlackBox mediante un código de colores.
- **Flow Chart**. Visualiza los resultados del BlackBox mediante un diagrama de flujo, cuyos componentes se relacionan mediante un código de color con los elementos del Rich Graphical Interface y BlackBox.
- **Pseudocode**. Presenta la codificación del algoritmo generado en el Flow Chart con sus correspondientes indicadores de colores.

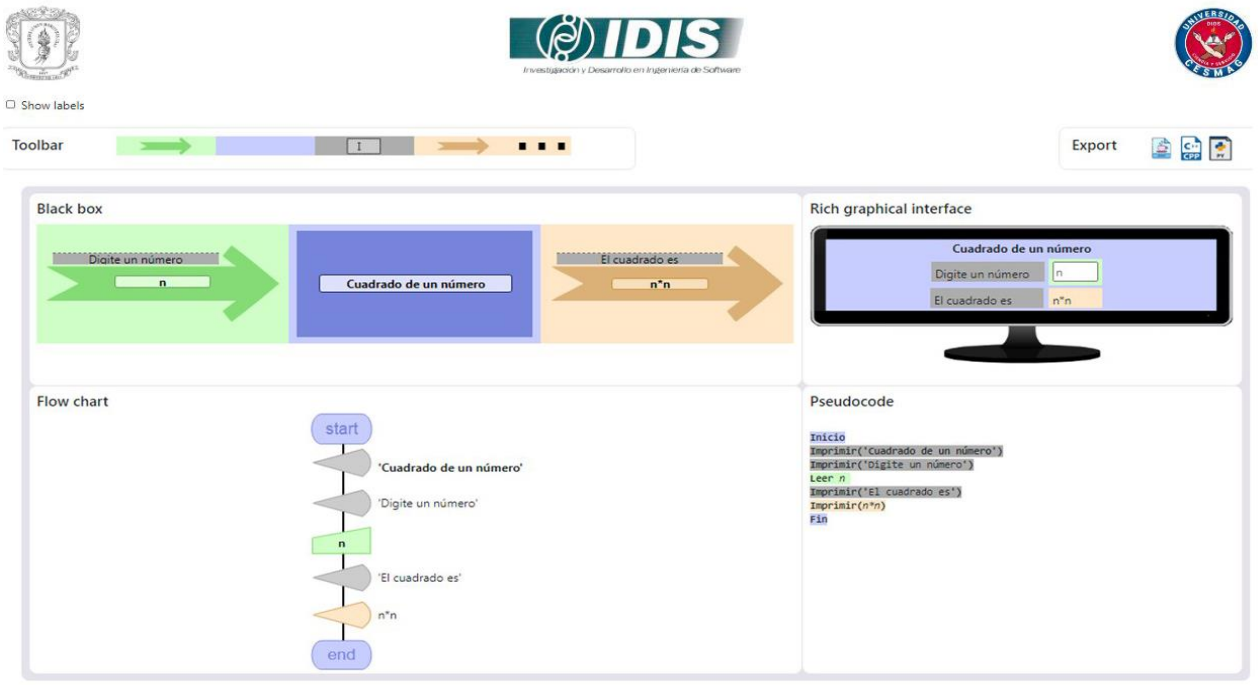


Figura 27. Entorno CodES.
Fuente: elaboración propia

Debido a que CodES está diseñado pensando en el desarrollo de habilidades algorítmicas para estudiantes de un CS1, consta de etiquetas que le permiten enriquecer la interfaz de usuario con el propósito de obtener entornos de interacción apropiados desde los primeros programas computacionales, asegurando de esta manera involucrar elementos básicos pero importantes a la hora de tener un producto software que incluyen un título del aplicativo hasta rótulos que guíen tanto la captura como la impresión de datos, que generalmente se descuida en los primeros ejercicios de programación.

CodES no contempla la operación de asignación por lo que es necesario realizar las operaciones aritméticas en las salidas del BlackBox, esto le permite al estudiante asociar cada proceso con su correspondiente elemento y así fundamentar el desarrollo inicial del pensamiento algorítmico.

Asimismo, CodES plantea una clave de identificación para enseñar el concepto de condicional, el cual se basa en que al tener una salida con dos alternativas de solución se debe pensar en la utilización de un condicional compuesto, como lo presenta la Figura 28.

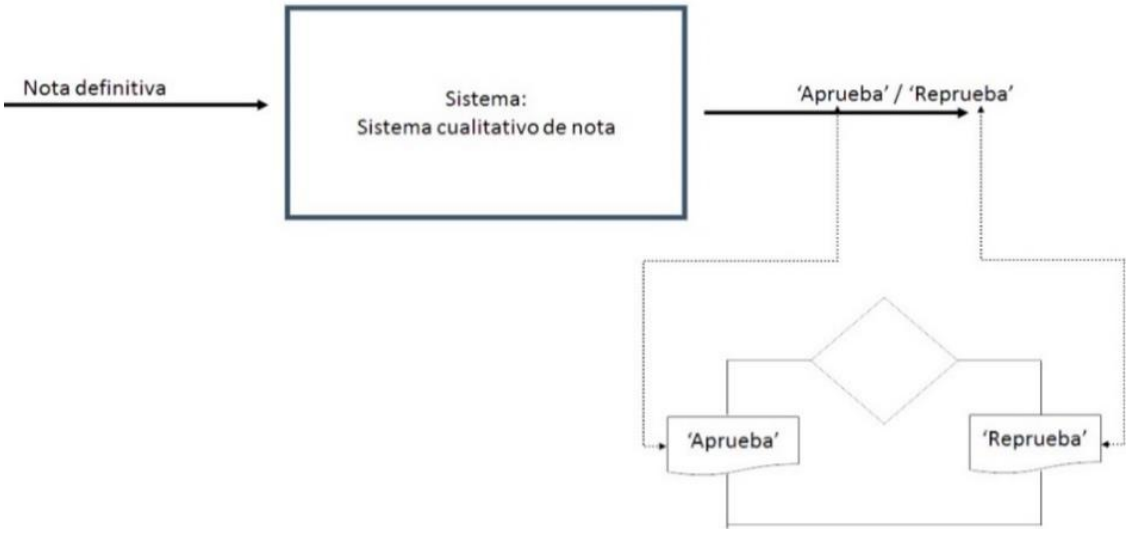


Figura 28. Condicional compuesto.
Fuente: elaboración propia

Además, si una salida tiene más de dos alternativas de solución (n), se está ante un condicional compuesto donde n-1 será la cantidad de condiciones requeridas (Figura 29), claro está sin la utilización de operadores lógicos como el And o el Or.

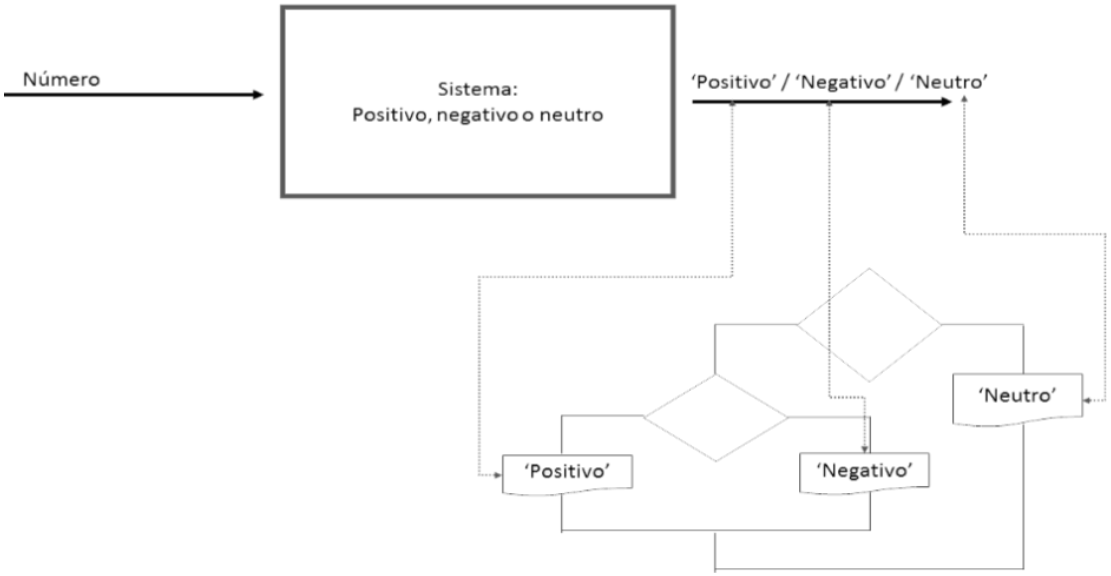


Figura 29. Condicional anidado
Fuente: elaboración propia

La Figura 30 presenta la implementación con CodES del ejemplo contenido en la Figura 29, teniendo en cuenta que el estudiante solo construye el BlackBox correspondiente y por cada instrucción se realiza la implementación respectiva en Rich Graphical Interface, Flow Chart y Pseudocode de forma automatizada.

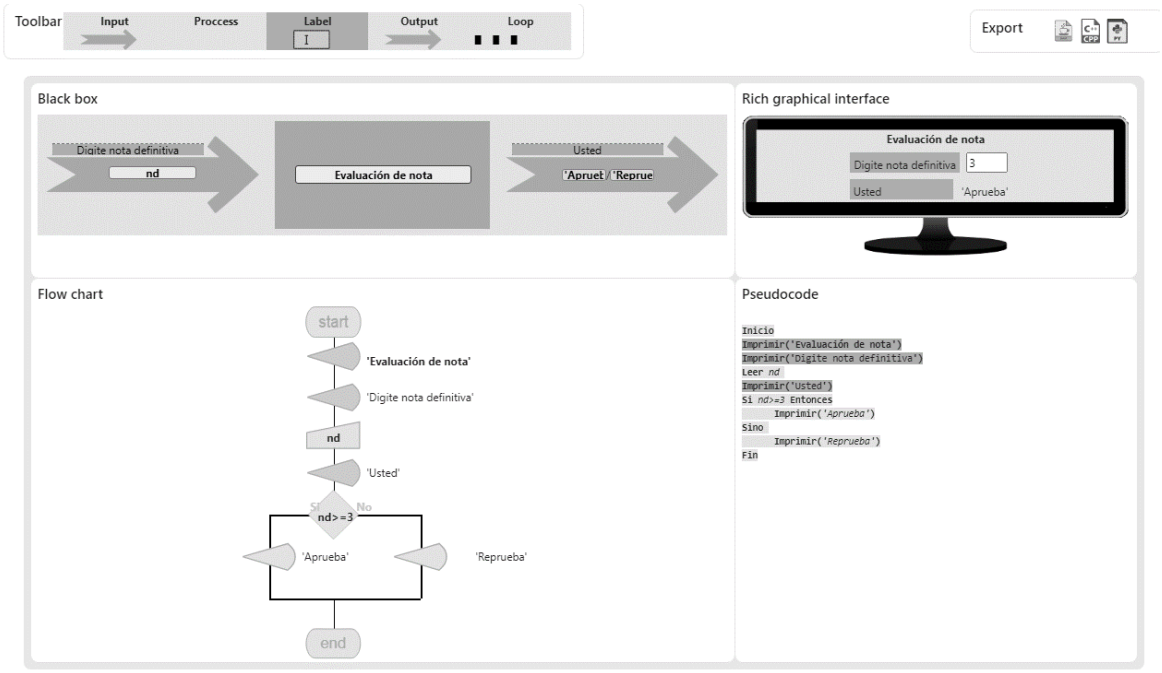


Figura 30. Condicional en CodES
Fuente: elaboración propia

Para la enseñanza de estructuras iniciales repetitivas conocidas como ciclos, CodES plantea la utilización del elemento continuidad representado con los tres puntos suspensivos, a través del cual en el BlackBox se plantea un diseño con al menos 4 elementos: una primera salida que indica el inicio, una segunda salida con la cual se calcula los pasos, el elemento de continuidad y finalmente una salida que determina el final, con estos elementos es posible identificar la estructura iterativa la cual puede ser visualizada en CodES mediante Ciclo Para, Mientras o Hacer mientras.

Además, se plantea una nueva extensión del concepto de ciclo de acuerdo a su estructura, para ello se propone la utilización del ciclo ascendente y descendente. El ciclo ascendente se caracteriza por a) el valor de inicio es mayor que el valor de fin, b) la condición o fin únicamente permite la utilización de los operadores relacionales menor o menor igual, c) los pasos son positivos; mientras que en el ciclo descendente a) el valor de inicio es menor que el valor de fin, b) la condición o fin únicamente permite la utilización de los operadores relacionales mayor o mayor igual y c) los pasos son negativos.

Una vez construido el BlackBox, CodES le permite al usuario evaluar dichas construcciones a través del Rich Graphical Interface, el cual permite capturar valores en las entradas, procesarlas y generar resultados. Asimismo, CodES se ejecuta Online e incorpora un verificador estático que le permite al estudiante corregir posibles errores al

validar la escritura de variables y operaciones automáticamente y a la vez posee una impresora estética que utiliza código de colores y distribución de código adecuada tanto en BlackBox, Rich Graphical Interfaz, Flow chart y Pseudocódigo.

3.2.2 CodES: Infraestructura tecnológica

CodES fue desarrollado con PhP, Java Script y MySQL. Es un software de visualización que incorpora herramientas de análisis propias de programas generadores de código fuente, que permiten guiar al usuario en implementación de sus proyectos. Entre las herramientas de análisis de código que emplea CodES se encuentran: Editor de estructura, impresora estética y verificador estático.

3.2.2.1 Editor de estructura. CodES identifica los elementos ubicados en el BlackBox y de acuerdo a su naturaleza, los convierte en estructuras de visualización en el RGI o en estructuras algorítmicas en el Flow Chart o en lenguaje de descripción algorítmica en el Pseudocode. Por ejemplo, ante un símbolo de “Entrada” en el BlackBox, CodES crea un TextBox en el RGI y lo ubica adecuadamente en el espacio asignado junto con una etiqueta para complementar su accionar. Asimismo, dicho símbolo de entrada crea gráficamente el símbolo de input en el algoritmo que se visualiza en el Flow Chart y le asigna la estructura sintáctica adecuada para su presentación. Finalmente, este símbolo de entrada también general la sintaxis del comando “Leer” con sus elementos estructurales en el la opción de Pseudocode.

3.2.2.2 Impresora estética: CodES implementa un código de colores para cada uno de las herramientas disponibles en el ToolBar, con el propósito de guiar al usuario en la identificación de cada herramienta diseñada en el BlackBox y su correspondiente equivalencia en el RGI, Flow Chart y Pseudocode. Por ejemplo, ante un símbolo de “Entrada” se dibuja una flecha de color verde en el BlackBox (), cuya equivalencia en el RGI es un TextBox () con su contorno en color verde, un símbolo de captura () en el Flow Chart y el comando Leer en el Pseudocode también de color verde.

3.2.2.3 Verificador estático. CodES tiene la capacidad de descubrir potenciales errores sin la necesidad de pasar por un proceso de compilación. Básicamente valida dos acciones: en primer lugar, lo hace con las herramientas del ToolBar, ya que no permite su ubicación en lugares que no son adecuados, esto es, por ejemplo, un entrada nunca podrá ubicarse en un proceso central o en una salida.

En segundo lugar, CodES implementa expresiones regulares para realizar validaciones en tres elementos: nombres de variables, salidas y condicionales.

3.2.2.3.1 Expresión regular para evaluar el nombre de las variables

CodES utiliza la siguiente expresión regular para validar el nombre de una variable (ver Tabla 7):

Alfabetos			
m={a,b,…,z}	n={A,B,C,…,Z}	b={ _ }	s={$}
Operadores			
U → Unión	( ) ( ) → Concatenación		* → Cero o más veces
Expresión Regular			
(m U n U b U s) (m U n U b U s)*			

Tabla 7. Expresión regular nombre de variables
Fuente: elaboración propia

La expresión regular presentada en la tabla 7, evalúa el nombre de las variables, el cual debe iniciar con un carácter que puede ser una mayúscula de ‘A’ a la ‘Z’ o una minúscula de la ‘a’ a la ‘z’ o uno de los dos caracteres especiales bien sea el guion bajo ‘_’ (underscore) o el símbolo de dólar ‘$’. En seguida, se puede utilizar una combinación de cero o varios caracteres que pueden ser dígitos del 0 al 9 o una mayúscula o una minúscula o uno de los dos caracteres especiales bien sea el guion bajo o el símbolo de dólar, como lo muestra la Figura 31.

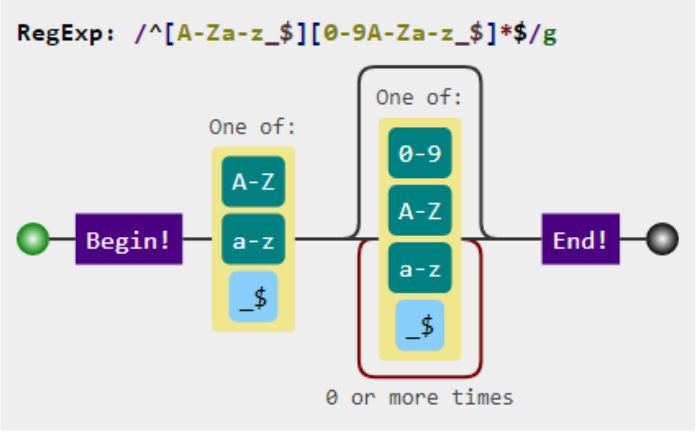


Figura 31. Expresión regular para nombre de variables
Fuente: elaboración propia

3.2.2.3.2 Expresión regular para evaluar una estructura de salida

CodES utiliza la siguientes expresiones regulares para validar una estructura de salida (ver Tablas 8 y 9):

Alfabetos			
d={0,1,…,9}	m={a,b,…,z}	n={A,B,C,…,Z}	
s={¿,?,!, ,_,,}	o={+,-,*,/ }	t={\t,\r,\n,\f,\}	c={‘}
Operadores			
U → Unión	**( ) ( )** → Concatenación	$^{+}$ → Uno o más veces	
Expresión Regular 1:			
c (d **U** m **U** n **U** s **U** o **U** t **)$^{+}$ c**			

Tabla 8. Expresión regular para escritura de rótulos
Fuente: elaboración propia

La expresión regular 1 de la tabla 8 evalúa la escritura de rótulos de salidas. El primer tipo de estas salidas se observa en la Figura 32 y el grupo 3, que exige el inicio con comilla simple seguido de una combinación de uno o varios caracteres que pueden ser dígitos del 0 al 9 o una mayúscula de la ‘A’ a la ‘Z’ o una minúscula de la ‘a’ a la ‘z’ o uno de los dos caracteres especiales como: el signo de interrogación de apertura ‘¿’ y cierre ‘?’, admiración ‘!’, espacio en blanco, guion bajo ‘_’; símbolos de operaciones aritméticas como: división ‘/’, multiplicación ‘*’, resta ‘-‘ y suma ‘+’; elementos de formato como: , la coma ‘,’, tabulación ‘\t’, retorno de carro sin avance de línea ‘\r’, salto de línea ‘\n’, avance de página ‘\f’, carácter de escape ‘\’ y comilla simple ‘’’ para finalizar.

Alfabetos			
v={n1,n2}	d={0,1,…,9}	o={+,-,*,/ }	
Operadores			
U → Unión	**( ) ( )** → Concatenación	* → Cero o más veces	$^{+}$ → Uno o más veces
Expresión Regular 2:			
((v **U** d^{+} **U** d*. d^{+}**)** **(**o**)** **(** v **U** d^{+} **U** d*. d^{+}**))$^{+}$**			

Tabla 9. Expresión regular para operaciones aritméticas con variables
Fuente: elaboración propia

La expresión regular 2 de la Tabla 9, permite construir una operación aritmética a partir de las entradas ya nombradas como variables, dichos nombres de variables se encuentran resaltados en verde, lo que indica que estos valores son dinámicos, es decir, varían según la cantidad de variables y el nombramiento que el usuario haga a las mismas. En el grupo 5 de la Figura 32 se observa que se evalúa el uso del nombre de una de las entradas, o también es posible escribir un número que tiene parte entera y también puede tener parte decimal, descrita en el grupo 7, seguido a ello, se debe agregar un símbolo aritmético contemplado en el grupo 9 y a este símbolo debe acompañarlo el nombre de una variable o un numero de la misma forma descrita, estos elementos pueden combinarse para realizar una composición de varias operaciones aritméticas con números y/o variables, como se muestra en la Figura 32.

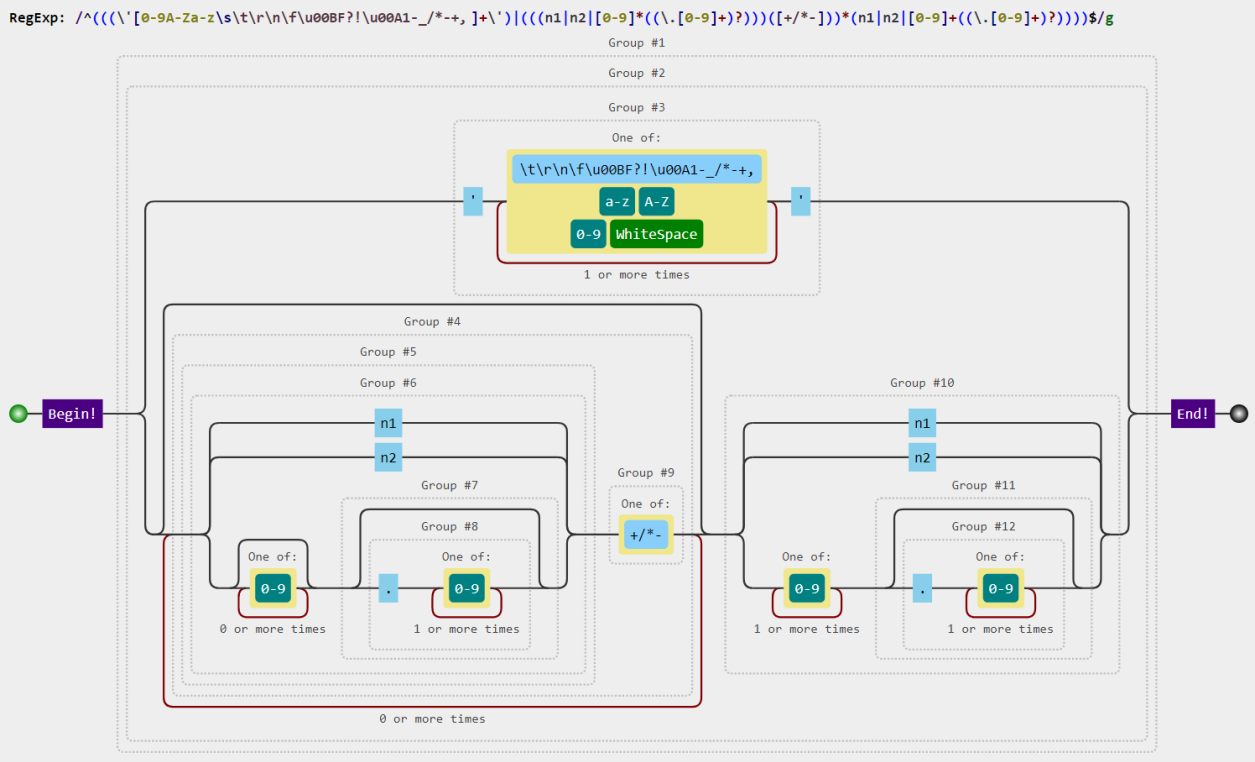


Figura 32. Expresión regular para estructuras de salida

Fuente: elaboración propia

3.2.2.3.3 Expresión regular para evaluar una estructura condicional

CodES utiliza la siguiente expresión regular para validar una estructura condicional (ver Tabla 10):

Alfabetos			
v={n1,n2}	d={0,1,…,9}	o={+,-,*,/ }	s={<,>,<=,>=,==}
Operadores			
U → Unión	( ) ( ) → Concatenación	* → Cero o más veces	+ → Uno o más veces
Expresión Regular:			
[(v U d+ U d*.d+) U ((v U d+ U d*. d+) % (v U d+ U d*.d+))] (s) (v U d+ U d*.d+)			

Tabla 10. Expresión regular para estructura condicional
Fuente: elaboración propia

La expresión regular de la Tabla 10, permite evaluar la correcta escritura de una operación de comparación de dos términos, donde en el término izquierdo se evalúa que contenga el nombre de una variable, dichos nombres de variables se encuentran resaltados en verde, lo que indica que estos valores son dinámicos, es decir, varían según la cantidad de variables y el nombramiento que el usuario haga a las mismas, o un número que puede ser entero o contener parte decimal, o puede realizarse la operación módulo % entre variables o números, seguido a esta expresión se evalúa una operación de comparación, las cuales se encuentran en el grupo 12, teniendo las operaciones de menor que “<”, mayor que “>”, menor o igual “<=”, mayor o igual “>=” o igualdad “==” y en el segundo término se valida la escritura de una variable o un número, como lo muestra la Figura 33.

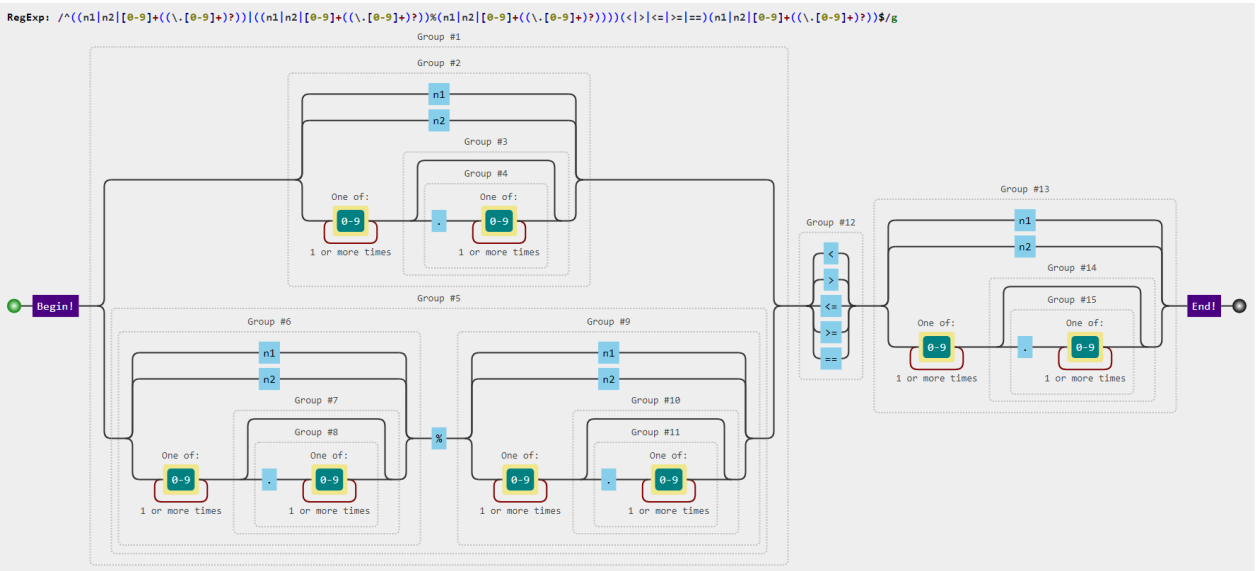


Figura 33. Expresión regular para estructuras condicional
Fuente: elaboración propia

3.2.2.4 Diagrama de clase de CodES. El diagrama de clase de CodES (Figura 34) muestra el módulo principal (main_js) el cual tiene dependencias directas con JQuery a través de las clase RegExp, que permite la evaluación de expresiones regulares y la clase Array, implementada para administrar las diferentes colecciones dentro del módulos. Adicionalmente, se observa la invocación a funciones propias de los objetos como lo es count() y countTo(). Asimismo, en el módulo main_js, se muestran los atributos que permiten administrar la generación y evaluación de los diagramas de flujo construidos por el usuario, también están los métodos que validan las interacciones del usuario y generan el código HTML, el cual es adicionado en el DOM**.**

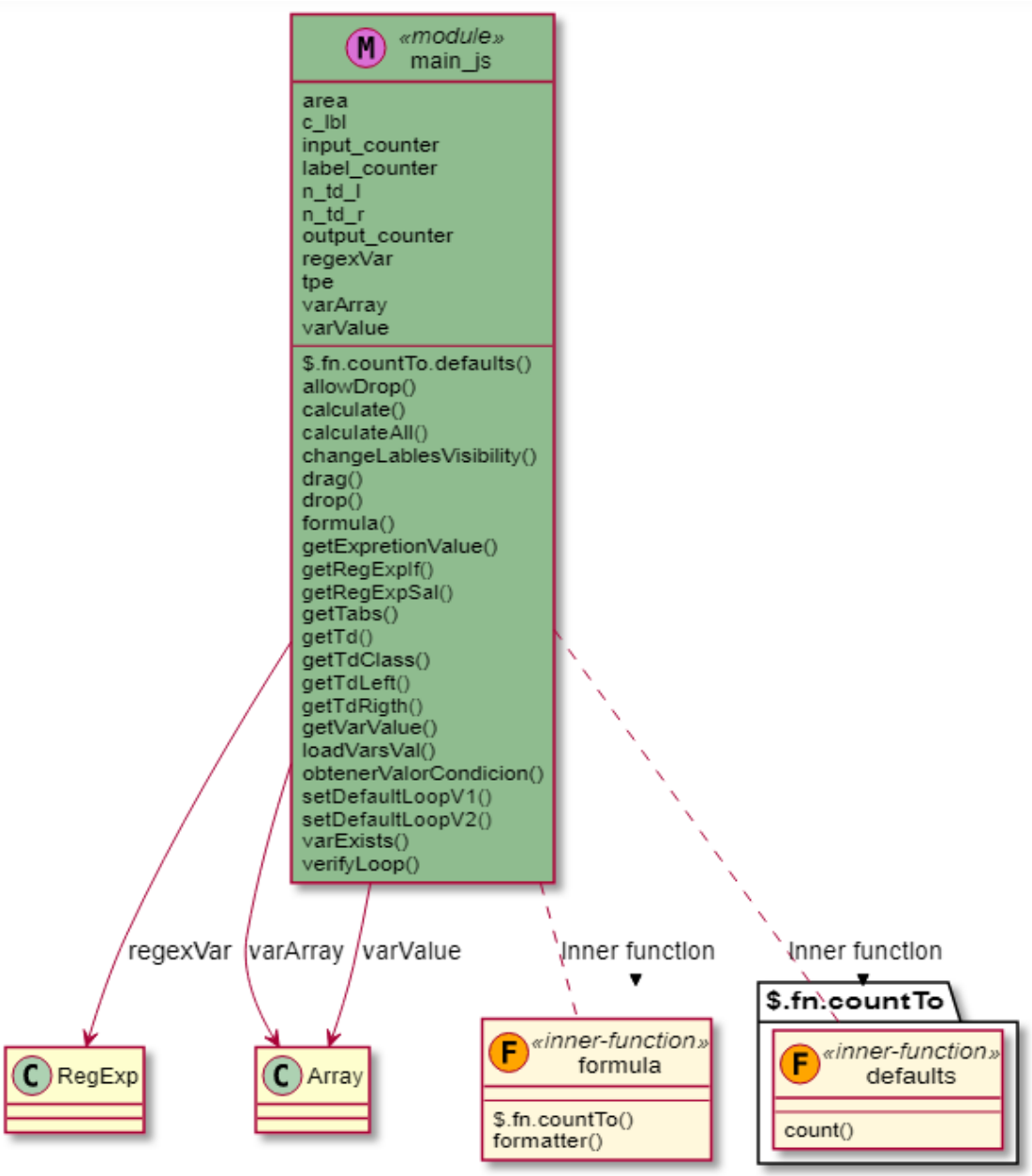


Figura 34. Diagrama de clases CodES

Fuente: elaboración propia

Capítulo 4

Resultados obtenidos

4.1 Resultados para modelo de descubrimiento de conocimiento

En esta sección se presentan los resultados obtenidos en cada etapa del modelo de descubrimiento de conocimiento, propuesto para construir analogías para la enseñanza de los conceptos fundamentales para desarrollar pensamiento algorítmico en los estudiantes universitarios en un CS1.

4.1.1 Etapa de procesamiento inicial

En el modelo de descubrimiento se utilizó un cuestionario para recolectar las analogías que los profesores utilizan en el aula de clase de forma planeada o espontánea en un CS1 en programas académicos universitarios. Dicho cuestionario recolectó información de datos generales de caracterización del profesor, universidad, curso y las analogías utilizadas en la enseñanza para cuatro conceptos fundamentales: entradas, salidas, condicionales y ciclos. La aplicación de esta encuesta reportó un total de 570 ejemplos de analogías en 15 universidades de 5 países de Centroamérica, Suramérica y Europa (ver Tabla 11) con un total de 33 profesores expertos (quienes realizaron el consentimiento informado de participación en este estudio, Anexo 5) en CS1 en programas universitarios de computación que forman profesionales en construcción de software.

País	Universidad	Tipo
Colombia	Universidad CESMAG	Privada
	Universidad del Cauca	Pública
	Universidad de Nariño	Pública
	Universidad Mariana	Privada
	Universidad Nacional Abierta y a Distancia	Pública
	Universidad Antonio Nariño	Privada
	Universidad Autónoma de Occidente	Privada
	Universidad San Buenaventura	Privada
	Corporación Universitaria Comfacauca	Privada
	Corporación Universitaria Autónoma de Nariño	Privada
Argentina	Universidad Nacional de Santiago del Estero	Pública
Ecuador	Universidad Politécnica Estatal del Carchi	Pública
México	Universidad Autónoma de Zacatecas	Pública
	Tecnológico Nacional de México / IT Aguascalientes	Pública
España	Universidad de Islas Baleares	Pública

Tabla 11. Universidades participantes en estudio

Fuente: elaboración propia

Con la recolección de datos concluida, se inició con las actividades de extracción que permitió evidenciar que de los 33 expertos 20 son de universidades públicas y 13 de universidades privadas (Anexo 6). Además, el 21% tienen formación doctoral, 70% con maestría y 9% como especialistas. Asimismo, en la caracterización de cursos, se pudo apreciar que los nombres con mayor frecuencia son: Introducción a la programación, Algoritmos y programación, Programación I y Fundamentos de programación que corresponde al 82% del nombre de los cursos. Así también, en la caracterización de las unidades académicas correspondientes con dichos cursos se encuentran bajo la denominación de Ingeniería de Sistemas, Computación e Ingeniería Informática con un 95%. Finalmente, todos los cursos consultados se orientan en el primer semestre de cada programa académico.

La actividad de pre-procesamiento termina con la construcción del Dataframe que contó con las siguientes variables: Nombre de la universidad, país donde se encuentra la universidad, tipo de universidad, último estudio del profesor, nombre del curso, semestre en el cual se imparte, programa o facultad, descripción de la analogía, concepto que pretende enseñar y si fue utilizada directamente en clases. La Tabla 12 muestra algunas analogías descritas por los profesores en esta etapa.

Analogía	Concepto
Ingresa un alimento para preparar comida	Entrada
En un ATH cuando se digita la clave	
Al momento de imprimir una factura	Salida
Calcula el total a pagar	
Al Escoger el menú del día	Condicional
Evalúa si es mayor o menor de edad	
Cuando se genera los diez primeros números	Ciclo
Registra productos en la caja registradora	

Tabla 12. Algunas analogías usadas en un CS1.
Fuente: elaboración propia

4.1.2 Etapa de descubrimiento

4.1.2.1 Transformación

Con los resultados de las actividades de pre procesamiento y transformación se construyó un modelo relacional de bases de datos (implementado en MySQL) con las entidades: Países, Universidades, Tipos_Universidades, Departamentos, Cursos, Profesores, Analogías, Conceptos y Descriptores (Figura 35).

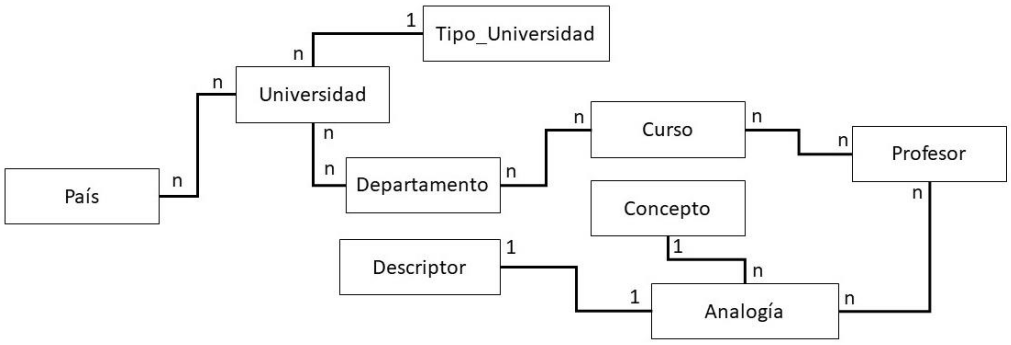


Figura 35. Modelo relacional de bases de datos
Fuente: elaboración propia

La actividad de transformación incorporó la creación de Datasets mediante vistas dinámicas que permitieron incluir los campos requeridos para la construcción de algoritmos de entrenamiento necesarios para la fase de minería.

Antes de realizar la actividad de minería se hace un pre análisis con NLTK con los primeros datos contenidos en el Dataset inicial, donde se obtuvo la siguiente información: cantidades de registros por cada Concepto: Entrada (57), salida (43), condicional simple (76), condicional compuesto (72), condicional anidado (64), selectiva múltiple (64), ciclo para (66), ciclo mientras (69) y ciclo hacer mientras (59) (Anexo 7).

Además, dicho Dataset (Anexo 8) contiene las siguientes particularidades: de los 570 casos de analogías empleadas por los profesores, 523 se han trabajado en clase con los estudiantes y 47 fueron propuestas debido a que el profesor no las ha utilizado para determinado concepto y sugiere se pueden utilizar. También se pudo observar que una misma analogía fue utilizada para explicar más de un concepto.

Asimismo, los contextos más utilizados en analogías para “entradas” fueron Alimentos, Compras y Recetas; para “salidas”: Producto, Facturas y Compras; para “condicional simple”: Clima, Compra y Vestido; para “Condicional compuesto”: Transporte, Edad y Supermercado; para “condicional anidado”: Compra, Salario y Vestido; para “estructura selectiva”: Calculadora, Comida y Cajero; para “ciclo para”: Múltiplos, Estudiantes y Reloj; para “ciclo mientras”: Estudiantes, Jugar y Múltiplos; y para “ciclo hacer mientras”: Juego, Estudiantes y Compras

También se encontraron 116 verbos diferentes y el de mayor frecuencia fue “Evaluar”; 309 sustantivos y el más empleado fue “Edad”; 402 acciones y la más frecuente fue “Evaluar edad”; 272 contextos y el de mayor concurrencia fue “Juego” y finalmente 11 categorías donde la de mayor repetición fue “Actividad”.

Estos hallazgos aplicados a un CS1 encuentran relación con los procesos de enseñanza-aprendizaje de la programación en estudios realizados en niños como lo presenta Pérez et al., [222][223], Sukamto y Megasari [199] o mediante metáforas por Chibaya [224].

4.1.2.2 Minería

4.1.2.2.1 Aprendizaje supervisado

La actividad de minería se inició con el tipo de aprendizaje supervisado mediante la tarea de clasificación, y con el propósito de obtener un modelo apropiado al conjunto de datos de entrenamiento. En esta tarea se escogió como clase el atributo concepto_general que categoriza las analogías para entradas, salidas, condicionales y ciclos; se emplearon 5 técnicas: Árboles de decisión, reglas de clasificación, máquinas de soporte vectorial, clasificador de envolturas y meta clasificadores. La implementación de estos métodos se realizó con Weka [225].

Para la tarea de clasificación por árboles de decisión, se emplearon dos métodos: J48 (que a su vez contiene el algoritmo de árboles de decisión C4.5, Anexo 9) que es uno de los más utilizados y a través de la configuración del nivel de confianza obtiene altas posibilidades de acierto en la capacidad de predicción del árbol construido y Logistic Model Trees (LMT, Anexo 9) que es un árbol de regresión logística que combina un modelo de árbol de decisión con un modelo de regresión logística en las hojas del árbol (con la que se predice el resultado de una variable) .

Debido al tamaño del Dataset, los árboles generados fueron complejos de analizar debido al número de hojas y tamaño del árbol, por ello, fue necesario utilizar tres técnicas de pre poda teniendo en cuenta de no afectar el umbral del porcentaje de instancias clasificadas correctamente. La primera técnica de poda consistió en manipular el factor de confianza en cada nodo (FC), la segunda mediante el número mínimo de instancias por hoja (IH) y la tercera la eliminación de atributos (EA).

Fue así como se planificó un conjunto de modelos de entrenamiento, aplicando para FC valores desde 10% hasta 40%; para IH desde 2 hasta 40 registros y EA hasta 3 negaciones del atributo principal, para un total de 48 árboles analizados. Además, también fue necesario un proceso de post-poda que consistió en descartar las reglas generadas que se encontraban bajo un umbral establecido de confianza y soporte de los datos.

Como resultado del proceso de pre poda, uno de los árboles de decisión generados que consideró atributos de acción, verbo, sustantivo y contexto, clasificó correctamente un 93,3% de las instancias con una confianza mínima de 83,9%. Este árbol tomó como nodo raíz el atributo verbo y las reglas generales de mayor valor se muestran en la Tabla 13:

Regla	Concepto	% de confianza	Registros por regla
Verbo=’ingresar’	Entrada	83,9%	31
Verbo=’obtener’	Salida	88,6%	21
Verbo=’evaluar’	Condicional	97,5%	193
Verbo=’hacer’	Ciclo	91,2%	131

Tabla 13. Pre poda con J48
Fuente: elaboración propia

Los métodos J48 y LMT en el proceso de prepoda clasificaron los siguientes verbos: Entrar, registrar, obtener, calcular, evaluar, hacer y jugar y los contextos aritméticos y de compra con las puntuaciones más altas. En el proceso de post-poda se evidencian variables importantes relacionadas con sustantivo, acción y contexto, algunos resultados fueron: Sustantivo (ingrediente, fila, plato, área, superior, asistencia, productos, presupuesto, etc.), acción (insertar, ingrediente, obtener, recibir dinero, evaluar edad, evaluar número, hacer múltiplos, jugar números, etc.) y contexto (receta, autobús, producto, fábrica, estudiantes, compras, juego, estudiantes, etc.).

Para la tarea de clasificación mediante reglas se obtuvieron 678 reglas y para ello se emplearon cuatro métodos:

Jrip (algoritmo Ripper, Anexo 9), que es un algoritmo eficiente que realiza procesos de poda incremental para reducción de errores y que obtiene reglas de clasificación que son optimizadas mediante técnicas heurísticas que utilizan un conjunto de reglas de entrenamiento específicas.

Modlem (Algoritmos de cobertura secuencial de Machine Learning, Anexo 9), el cual utiliza conjuntos mínimos de reglas para buscar aproximaciones de forma sucesiva en las clases de decisión.

OneR (One Rules es basado en ID3, Anexo 9) que es un algoritmo sencillo y preciso que construye una regla por cada conjunto de datos evaluados y selecciona la de menor error.

Part (Lista de decisiones Parte basado en C4.5, Anexo 9) que se caracteriza por construir un árbol de decisión auxiliar, convirtiendo la mejor hoja en una hoja en cada iteración.

A su vez, la clasificación mediante máquinas de soporte vectorial se realizó con el método SMO (Sequential Minimum Optimization, Anexo 9) que es un algoritmo secuencial de optimización mínima, utilizado especialmente para la categorización de textos, y mediante el cual se obtuvo un total de 1756 reglas.

En la clasificación mediante envolturas se recurrió al método InputMappedClassifier (Anexo 9) que utiliza los datos de entrenamiento y test que son no compatibles para construir un nuevo mapa con los datos de ensayo entrantes y los datos de entrenamiento, este algoritmo clasificó 21 reglas.

Además, utilizando el Meta clasificador IterativeClassifierOptimizer (Anexo 9) que es un clasificador iterativo que se caracteriza por elegir la mejor iteración evaluada, se realizaron 10 iteraciones con validación cruzada, obteniendo 40 reglas.

Finalmente, los resultados para el tipo de aprendizaje supervisado en la tarea de clasificación se obtuvieron teniendo en cuenta los indicadores de precisión, área bajo la curva, clasificación y la tasa positiva para cada uno de los métodos empleados. Para el concepto de entrada dichos valores se presentan en la Tabla 14.

Método	Entrada %	Salida %	Cond. %	Ciclo %
J48	98.4	98.4	98.4	98.4
LMT	94.4	94.4	94.4	94.4
Jrip	81.1	81.1	81.1	81.1
Modlem	99.1	98.9	99.1	99.1
OneR	98.6	98.6	98.6	98.6
Part	80.5	86.1	80.5	86.1
SMO	86.9	86.9	86.9	86.9
InputMappedClassifier	97.7	97.7	97.7	97.7
IterativeClassifierOptimizer	85.8	85.8	85.8	85.8

Tabla 14. Instancias correctamente clasificadas
Fuente: elaboración propia

A continuación, se presentan las reglas más importantes del aprendizaje supervisado para cada uno de los conceptos establecidos en este estudio.

Para el concepto de entrada, las reglas de clasificación más importantes obtenidas en los 9 métodos se muestran en la Tabla 15:

Regla	Verbo	Contexto	Sustantivo
1	ingresar	receta	ingrediente, alimento
2	ingresar, colocar	receta, salario, aritmética, estudiantil, precio	
3	registrar, digitar, leer	salario, aritmética, estudiantil, precio	

Tabla 15. Reglas de clasificación más importantes para “Entrada”
Fuente: elaboración propia

Para el concepto de salida, las reglas de clasificación más importantes obtenidas en los 9 métodos se presentan en la Tabla 16:

Regla	Verbo	Contexto	Sustantivo	Acciones
1	obtener	receta	ingrediente, alimento	
2	obtener, calcular		total, área, valor	obtener resultado
3	imprimir, calcular	aritmética		
4	generar, facturar, imprimir		factura	

Tabla 16. Reglas de clasificación más importantes para “Salida”
Fuente: elaboración propia

Asimismo, las reglas de clasificación más importantes para el concepto de condicional se exhiben en la Tabla 17.

Regla	Verbo	Contexto	Sustantivo	Acción
1	elegir	alimento, restaurante	comida, hora	
2	evaluar	matemático, estudiantil	edad, número, calificación, precio	
3	elegir, determinar, evaluar	aritmética		
4			*sustantivo	evaluar

Tabla 17. Reglas de clasificación más importantes para “Condicional”
Fuente: elaboración propia

Además, las reglas de clasificación más importantes para el concepto de ciclo se muestran en la Tabla 18:

Regla	Verbo	Contexto	Sustantivo	Acción
1	hacer, contar, repetir		edad, múltiplo	
2	sumar, promediar			sumar edades, sumar salarios, promediar edad
3	sumar, contar	aritmética, estudiantil		
4		compras, cajero, jugar, televisión	fila, dinero, factura, energía, turno	
5		jugar	ficha	
6	ensamblar, llenar		vehículo, fila, factura	

Tabla 18. Reglas de clasificación más importantes para “Ciclo”
Fuente: elaboración propia

4.1.2.2.2 Aprendizaje no supervisado

Se inició mediante la tarea de asociación a través de dos métodos:

Apriori, que es un algoritmo que encuentra eficientemente ítems frecuentes en un conjunto de datos, con los cuales construye reglas de asociación utilizando un enfoque top-down.

PredictiveApriori, es un algoritmo que obtiene las mejores reglas de acuerdo con un valor de confianza, utiliza un enfoque bottom-up.

En la generación de reglas con Apriori (Anexo 9) se utilizó un soporte mínimo del 1% con una confianza de al menos un 70%, en total se obtuvieron 328 reglas, las principales se muestran en la Tabla 19.

Concepto	Regla
Entrada	(verbo in {ingresar}) & (sustantivo in {ingrediente})
Salida	(verbo in {obtener, generar, imprimir})
Condicional	(verbo in {elegir, evaluar, determinar} & (sustantivo in {operación, numero, nota, compra, opciones, comida, clima, edad} or contexto in {compra, comida, calculadora, estudiante}))
Ciclo	(verbo in {hacer,jugar, contar, ensamblar, sumar} & (sustantivo in {múltiplos, vehículo} or contexto in {aritmética}))

Tabla 19. Asociación con A priori
Fuente: elaboración propia

En la búsqueda de reglas con PredictiveApriori (Anexo 9) se configuraron los mismos parámetros de soporte y confianza de Apriori, obteniéndose 361 reglas de asociación.

Luego, se complementó con la tarea de agrupamiento o segmentación, la cual permite que las diferencias intra grupales se minimicen y las extra grupales se maximicen.

Antes de aplicar esta técnica, se realizó con los datos un proceso de escalamiento, ponderación y selección para obtener un nivel de segmentación de mayor confiabilidad.

Para la tarea de agrupamiento se utilizó la técnica K-Means que reúne objetos en k grupos de acuerdo a sus características más relevantes, utilizando un problema de optimización para reducir las distancias entre las características agrupadas y una característica central (también llamada centroide), es decir, busca coincidencias entre los datos de manera iterativa, comparando siempre con uno de ellos a la vez.

Dicha técnica se configuró con un valor de 120 como semilla en la generación de números aleatorios y 4 para número de grupos. Para K-Means el agrupamiento para entradas, salidas, condicionales y ciclos de acuerdo a los datos clasificados en el Dataset, tuvo un acierto del 89%, 91%, 99% y 85% respectivamente.

Las reglas más importantes del aprendizaje no supervisado se presentan a continuación.

Los resultados más importantes para el tipo de aprendizaje no supervisado en la tarea de asociación para el concepto de entrada, se muestran en la Tabla 20:

Regla	Verbo	Contexto	Sustantivo
1	ingresar		ingrediente, clave
2	ingresar	compra	
3	leer, ingresar, digitar, insertar		precio, lado, clave, dinero, tarjeta, código
4	digitar, insertar		aritmética
5	leer, ingresar		estudiante
6	ingresar, entrar	comida	
7		estudiantil, dato	clave
8	ingresar, leer, ensamblar	receta, comida, compra	ingrediente, clave, parte
9	ingresar		material

Tabla 20. Reglas de asociación más importantes para “Entrada”

Fuente: elaboración propia

Asimismo, en la Tabla 21 se muestran las reglas más importantes en la tarea de asociación para el concepto de salida:

Regla	Verbo	Contexto	Sustantivo
1	obtener, generar, imprimir		
2	obtener, generar, recibir, terminar	factura, información	plato, producto, información
3	imprimir	factura, documento, aritmética, cálculo	
4	obtener, calcular, generar		área, valor, total, información
5	obtener	receta	ingrediente, alimento, bebida
6	obtener, generar, imprimir, calcular	aritmética	
7	obtener	receta	plato, bebida
8	imprimir, obtener		factura, documento, saldo, información, mensaje, área, foto, contacto, nombre
9	salir	transporte, fábrica	bus, lava-auto, fila

Tabla 21. Reglas de asociación más importantes para “Salida”

Fuente: elaboración propia

Igualmente, en la Tabla 22 de presentan las reglas más importantes para el concepto de condicional en la tarea de asociación.

Regla	Verbo	Contexto	Sustantivo
1	elegir, evaluar, determinar	compra, comida, calculadora, estudiante	operación, numero, nota, compra, opciones, comida, clima, edad
2	evaluar, elegir, determinar, validar	edad, clima, nota, compra, salario	número, nota, compra, comida, color, edad, clima, precio
3	elegir	restaurante	menú, comida
4	evaluar		edad, calificación, camino, número, precio, salario, peso, horario, ruta, rango, estado, plan, compras, opciones, clima, destino
5	evaluar	estudiantil, comida, acción, decisión, compras, fabricación, aritmética, juego, operación, salud, transporte	
6	elegir		camino, batería, caso, operación, camino, opción, opción, operación, música, operación, canción
7	evaluar		apagador, señal, identidad, circuito, resistencia, batería, alternativas, operación, circuito, autentificación, subsidio, sonidos, operaciones, parte, sala, operaciones, nota, producto, turno, respuestas, asignatura, opciones, respuestas, texto, asignatura
8	elegir	vestuario	prenda
9	comparar, evaluar, elegir	compras	

Tabla 22. Reglas de asociación más importantes para “Condicional”

Fuente: elaboración propia

Las reglas de ciclos más importantes para la tarea de asociación se presentan en la Tabla 23:

Regla	Verbo	Contexto	Sustantivo
1	hacer, jugar, contar, ensamblar, sumar	aritmética	múltiplos, vehículo
2	jugar, contar, hacer, sumar, recorrer	aritmética, estudiante, presupuesto	múltiplos, estudiante, producto
3	jugar		carrera_numérica, profesiones, parqués, balota, tingo_tango, escondite,

Regla	Verbo	Contexto	Sustantivo
			guerra_naval, cartas, cucunova, Hanói, adivinar_número
4	sumar, contar, promediar		edad, estatura, peso, salario, cosa, variable, número, compra, color, prenda, ingrediente, contacto
5	hacer		rutina, operación, ejercicio, pasteles, lectura, tejido, receta, disfraces, múltiplos, giros, Fibonacci, suma, flexiones, actividades, deberes, fila
6	hacer, sumar, ejecutar, contar, promediar, calcular, ejecutar	aritmética, cálculo	
7	agendar, hacer, controlar, subir, repartir, sellar, aflojar, corregir, imprimir, mover, lavar, preguntar, contar, caminar, consumir, identificar, realizar, programar, atender, seguir, tener, encender, tomar, desarrollar, conectar, sembrar, asistir, repetir, recolectar, entregar, jugar, arreglar, resolver, pintar, llamar, calificar, buscar, trabajar	actividad	

Tabla 23. Reglas de asociación más importantes para “Ciclo”
Fuente: elaboración propia

Por último, para el aprendizaje no supervisado en la tarea de agrupamiento o segmentación y con el fin de determinar características similares únicamente del concepto de entrada, se generó un nuevo Dataset solo con las tuplas correspondientes y al cual se descartó el atributo clase entrada, con ello, se configuró la técnica K-Means para particionarlos en 3 grupos. Asimismo, se realizaron 3 Dataset más con los idénticos criterios para los conceptos de salida, condicional y ciclo. Los resultados de aprecian en la Tabla 24.

Concepto	Variable	Cluster 0	Cluster 1	Cluster 2
Entrada	Verbo	Ingresar	Leer	Registrar
	Contexto	Receta	aritmética	compras
Salida	Verbo	Obtener Generar	Calcular Imprimir	Obtener

Concepto	Variable	Cluster 0	Cluster 1	Cluster 2
	Contexto	Aritmético Acción	Compras	Receta
Condicional	Verbo	Elegir	Evaluar	Determinar Elegir
	Contexto	Diversos	Diversos	Diversos
Ciclo	Verbo	Hacer Jugar	Contar	Hacer
	Contexto	Cálculos	Diversos	Actividades

Tabla 24. Agrupamiento por concepto K-Means
Fuente: elaboración propia

4.1.2.2.3 Aprendizaje semi supervisado

El algoritmo EM (Expectation Maximization) realiza un procedimiento probabilístico para encontrar el grupo de clústeres más similares, teniendo en cuenta un conjunto de puntuaciones calculadas en cada iteración.

EM fue configurada con el mismo valor de semilla de K-Means y a pesar que en EM (Anexo 9) no se configuró el número de clúster, este generó 4 de estos. Igualmente, los resultados más importantes para el tipo de aprendizaje semi supervisado en la tarea de agrupamiento o segmentación con la técnica EM y con la misma metodología realizada con K-Means con 4 Dataset se obtuvieron los resultados que se aprecian en la Tabla 25.

Concepto	Variable	Cluster 0	Cluster 1	Cluster 2
Entrada	Verbo	Diversos	Ingresar	Ingresar registrar
	Contexto	Actividad	Receta Dispositivos	Compras Dispositivo
Salida	Verbo	Obtener Calcular	Generar Imprimir	Diversos
	Contexto	Aritmético Receta	Acción Compra	Vehículo

Concepto	Variable	Cluster 0	Cluster 1	Cluster 2
Condicional	Verbo	Evaluar Calcular	Evaluar	Evaluar
	Contexto	Numérico	Actividad	Calculadora
Ciclo	Verbo	Hacer	Jugar	Contar Sumar
	Contexto	Numérico Actividad	Juegos	Numérico

Tabla 25. Agrupamiento por concepto EM
Fuente: elaboración propia

4.1.3 Etapa de análisis

Concluida la actividad de minería y utilizando NLTK se aplicaron tres labores: Análisis lingüístico, analítica textual y análisis experimental por cada uno de los cuatro conceptos principales establecidos en este estudio.

4.1.3.1 Análisis Lingüístico

Esta actividad se desarrolló con NLTK y tuvo como propósito la extracción de información del corpus de las analogías aquí analizadas para complementar los patrones obtenidos en la fase de descubrimiento.

Como resultado del análisis lingüístico en las analogías descritas por los profesores para afrontar la abstracción del concepto de entrada, se contó con una variedad léxica del 81% donde la utilización de verbos tiene un 12,2% y sustantivos 43,5%. Los principales resultados para los verbos, sustantivos, acciones y contextos más utilizados se muestran en la Tabla 26.

Verbos	Sustantivos	Acciones	Contextos
ingresar, registrar, leer, colocar, llenar, insertar, digitar, entrar e iniciar	ingrediente, lado, fila, alimento, objeto, clave, dato, dinero, articulo, tarjeta, código y día	ingresar ingrediente, ingresar clave, leer texto, ingresar bus, encender circuito, llenar casillero, hacer fila, llenar combustible, digitar dato	receta, bus, aritmética, estudiantil, dato, restaurante, receta, teléfono, salario y precio

Tabla 26. Principales resultados del análisis lingüístico concepto de Entrada
Fuente: elaboración propia

Además, en la Figura 36 se observa el proceso de etiquetación morfosintáctico (Anexo 10) para la analogía “Clave en un cajero automático”.

(T) Head position	(T) Head	(T) Head category	(T) Dependent position	(T) Dependent	(T) Dependent category	(T) Syntactic relation
0	clave	nombre	2	cajero	nombre	Complemento nominal
2	cajero	nombre	3	automático	adjetivo	Modificador

Figura 36. Etiquetación morfosintáctica entrada

De igual manera, en el análisis lingüístico en las analogías para enseñar el concepto de Salida, se contó con una variedad léxica del 71,2% donde la utilización de verbos tiene un 9,4% y de sustantivos un 25,9%. Los principales resultados para los verbos, sustantivos, acciones y contextos más utilizados se exhiben en la Tabla 27.

Verbos	Sustantivos	Acciones	Contextos
obtener, calcular, generar, imprimir y facturar	total, área, malteada, plato, información, encendido, malteada, ropa, resultado, factura, salarios, sonido y valor	obtener receta, recibir dinero, salir del bus, obtener solución, motor encendido, imprimir factura, obtener resultado aritmético, generar documento, imprimir documento y obtener plato preparado	producto, fábrica, documento, horno, aritmética, impresión y factura

Tabla 27. Principales resultados del análisis lingüístico concepto de Salida
Fuente: elaboración propia

En los resultados del análisis lingüístico en las analogías utilizadas para el estudio de condicionales, existió una variedad léxica del 61,6% donde la utilización de verbos fue de 7,8% y sustantivos 16,9%. Los principales resultados para los verbos, sustantivos, acciones y contextos más utilizados se presentan en la Tabla 28.

Verbos	Sustantivos	Acciones	Contextos
evaluar, elegir, determinar, validar, tomar, comparar, y calcular	mayor, menor, asistencia, ruta, operación, transporte, camino, calificación, prenda, compra, opciones, salud, color, señal, alternativas, material, actividad, hora, precio, alternativas, peso y asignatura	evaluar edad, evaluar número, evaluar nota, evaluar compra, elegir transporte, elegir camino, determinar edad, determinar género, calcular mayor, calcular menor, elegir caso, evaluar nota, evaluar cantidad, evaluar color, evaluar clima, elegir prenda, elegir destino, y evaluar número	estudiantil, compras, aritmética, y transporte

Tabla 28. Principales resultados del análisis lingüístico concepto de Condicional
Fuente: elaboración propia

Asimismo, los resultados lingüísticos para el estudio de ciclos, se contó con una variedad léxica del 46,2% donde la utilización de verbos fue de 6,4% y de sustantivos de 12,1%. Los principales resultados para los verbos, sustantivos, acciones y contextos más utilizados se presentan en la Tabla 29.

Verbos	Sustantivos	Acciones	Contextos
hacer, jugar, ensamblar, contar sumar, realizar, recorrer, ejecutar, contar, promediar, determinar, repetir, enjuagar, recolectar, poner y convertir	producto, usuarios, aritmética, presupuesto, conteo, actividad, juego, carta, edad, múltiplo, rutina y Fibonacci	hacer múltiplos, jugar números, ensamblar vehículos, contar estudiantes, hacer rutinas, hacer operaciones, sumar edades, sumar salarios, controlar ingreso, subir pisos, contar estudiantes, promediar edad, solicitar datos personales y jugar	juego, estudiantil, aritmética, presupuesto, tareas, nomina, cocina, salud, lectura, deporte, conteo y ensamblar

Tabla 29. Principales resultados del análisis lingüístico concepto de Ciclo
Fuente: elaboración propia

4.1.3.2 Analítica textual

Los principales resultados para la analítica textual para el concepto de entrada, mostraron que las palabras clave más destacadas son: ingrediente, digitado, dispensador e ingresar. Las multi-palabras de mayor frecuencia: Receta de cocina, preparación de alimento, geometría figuras, altura lado, dispensador bebidas, asuntos en bancos, cantidad de artículos y clave cajero. Por su parte, en el análisis de entidades de las analogías reportadas se encontró que ingrediente y geometría son los más frecuentes. El analizador de sentimientos indicó que de los ejemplos planteados el 17,9% son frases negativas, el 57,1% son frases neutrales y 25,0% son frases positivas.

Asimismo, en la analítica textual para el concepto de Salida, los principales resultados de palabras clave fueron: plato, aritmético, operación y obtener. Las multi-palabras de mayor frecuencia: Receta preparada, figuras geométricas, recibo, operación aritmética, calcular salario, mensaje informativo, cantidad de artículos, volumen de figuras y dinero de cajero. En el análisis de entidades de mayor presencia en las analogías fueron: Área y concluido. El analizador de sentimientos indicó que de los ejemplos planteados el 15,9% son frases negativas, el 54,5% son frases neutrales y 29,6% son frases positivas.

También, los principales resultados de la analítica textual para condicionales en cuanto a palabras claves fueron: Si, menú, semáforo, casos, validación y evaluar. Las multi-palabras de mayor frecuencia: elección de prenda, evaluar opción, peso ideal, vida cotidiana, elegir destino, tipo de transporte, menú restaurante y evaluar propuesta. En el

análisis de entidades de mayor presencia en las analogías fueron: Menú, prendas y ciudades. El analizador de sentimientos indicó que de los ejemplos planteados el 17,2% son frases negativas, el 32,2% son frases neutrales y 50,6% son frases positivas.

Igualmente, los principales resultados de la analítica textual en las analogías utilizadas para ciclos, las palabras claves fueron: Hacer, sumatoria, factorial, torres Hanói, iterar, contar, jugar y mientras. Las multi-palabras de mayor frecuencia: carrera numérica, juego de carrera, piso de edificio, promedio de edad, compañeros de curso, cajero supermercado, línea de ensamblaje y hacer múltiplos. En el análisis de entidades de mayor presencia en las analogías fueron: Números, Fibonacci, iterar y sumatorias. El analizador de sentimientos indicó que de los ejemplos planteados el 17,5% son frases negativas, el 41,8% son frases neutrales y 40,7% son frases positivas.

4.1.3.3 Datos enlazados

En los resultados de extracción de datos enlazados mediante tripletas (Anexo 11) para el concepto de entrada, las relaciones entre sujeto y objeto más importantes fueron: Se usarán para, ingresa, representa, requiere, entra, como lo muestra la Tabla 30.

Sujeto	Relación	Objeto
Ingredientes	Se usarán para	Una comida
Cliente, persona, aritmética	Ingresa	Valor
Número	Representa	Valor en una calculadora
La harina	Requiere	Un panadero para hacer pan
Dispensador de bebidas	Entra	dinero

Tabla 30. Tripletas

Fuente: elaboración propia

De igual manera, en la extracción de tripletas en el concepto de salida, las relaciones entre sujeto y objeto más importantes fueron: obtiene un, produce una, sale.

Asimismo, en las tripletas para condicionales las relaciones entre sujeto y objeto de mayor frecuencia fueron: si está, si cumple, si es, ser y viajar en.

Finalmente, en la extracción de datos enlazados para ciclos, las relaciones más relevantes fueron: se repiten según, hacen, sumar, hasta y mientras sean.

4.1.4 Hallazgos del modelo de descubrimiento de conocimiento

Se puede apreciar en la gran mayoría de las analogías recolectadas en este estudio, que los profesores las redactan incluyendo verbos clave usados comúnmente en la terminología computacional y las combinan con contextos y acciones que forman parte del accionar cotidiano de las personas, donde recetas de cocina, cálculos aritméticos, compras, ventas, situaciones escolares, tareas del hogar, juegos, entre otros, son los más utilizados.

Además, se aprecia que los profesores utilizan en mayor proporción analogías que pueden ser modeladas computacionalmente en lugar de aquellas que son generales y que no pueden convertirse en programas de computador.

A continuación, se presenta con mayor detalle la discusión por cada uno de los tres tipos de aprendizaje contemplados en este estudio

En el aprendizaje supervisado, se puede ver que los profesores incorporan verbos como ingresar, leer, escribir y registrarse para representar una acción de entrada, además, los ejemplos presentados se desarrollan de manera repetitiva en el contexto de recetas de cocina, ejercicios básicos de nómina, cálculo de operaciones aritméticas, procesamiento de calificaciones en el ambiente estudiantil, transacciones en compras, acciones en dispositivos electrónicos, entre otros.

En estas reglas se puede afirmar también, que los profesores incluyen palabras clave en el uso de verbos como obtener, calcular, generar, facturar e imprimir que son de uso común en terminología computacional para denotar una salida. Además, los contextos de los ejemplos se enmarcan en la obtención de una receta de cocina, el cálculo o impresión de nóminas, operaciones aritméticas, geométricas, facturas, etc. También existen salidas que representan acciones como la obtención de una combinación adecuada de prendas, coches y ropa, estímulos y resultados de procesos de dispositivos electrónicos.

En las reglas encontradas para las analogías utilizadas para la abstracción del concepto de condicional, también se observa que los profesores incluyen verbos característicos de la terminología específica como elegir, evaluar, determinar y comparar. Los contextos analizados se presentan a la hora de elegir una comida de un menú, la evaluación de conceptos matemáticos, La evaluación de aspectos académicos del

estudiante y en actividades como hacer las tareas del hogar, cruzar la calle, visitar, viajar, comer, bailar, etc.; elección de ropa, etc.

Finalmente, las reglas generadas en las analogías utilizadas para enseñar el concepto de ciclo también incorporan verbos que están estrechamente relacionados con la terminología específica al momento de presentar un ejemplo computacional para este concepto, entre ellos están: Hacer, contar, repetir, sumar y promediar. También se registran acciones como: Sumas, promedios, conteos de edad, salarios, múltiplos, pasajeros, pasos, estudiantes, entre otros.

En el aprendizaje no supervisado, para el concepto de Entrada, las reglas de asociación obtenidas están directamente relacionadas con las extraídas con las técnicas de aprendizaje supervisado, donde el uso de verbos juega un papel importante en la oración propuesta y con mayor énfasis en sustantivos en contextos similares.

Las reglas de asociación para el concepto de salida son más específicas que las obtenidas en la tarea de clasificación y estas utilizan principalmente los mismos verbos y contextos. Además, existen reglas que involucran acciones en contextos generales y no puramente computacionales.

Las reglas de asociación para el concepto de condicional son más detalladas que las generadas por técnicas de clasificación con mayor participación de sustantivos y contextos, pero compartiendo los mismos verbos.

Las reglas de asociación encontradas en las analogías utilizadas para la enseñanza de ciclos contienen tanto los mismos verbos, sustantivos y contextos como las reglas de clasificación, pero estas incluyen más descriptores para verbos y sustantivos, lo que nos permite contemplar un mayor número de posibilidades a la hora de analizar potenciales contextos analógicos para abordar este concepto con nuevos ejemplos.

En el aprendizaje semi supervisado, los resultados de la tarea de agrupación de las Tablas 24 y 25 reafirman las principales reglas obtenidas tanto en la tarea de clasificación como en la de asociación previamente analizadas y a pesar de realizar agrupaciones internas en cada concepto, subgrupos que utilizan los mismos verbos y contextos ya fueron encontrados.

De acuerdo con los resultados encontrados en el aprendizaje supervisado, no supervisado, semi supervisado, análisis lingüístico, analítica textual y datos enlazados, en esta sección se presenta un modelo (Figura 37) que integra el aprendizaje situado y el modelo didáctico analógico para tener un mayor acercamiento a los conceptos

fundamentales de programación a enseñar por parte del profesor para el desarrollo de pensamiento algorítmico.

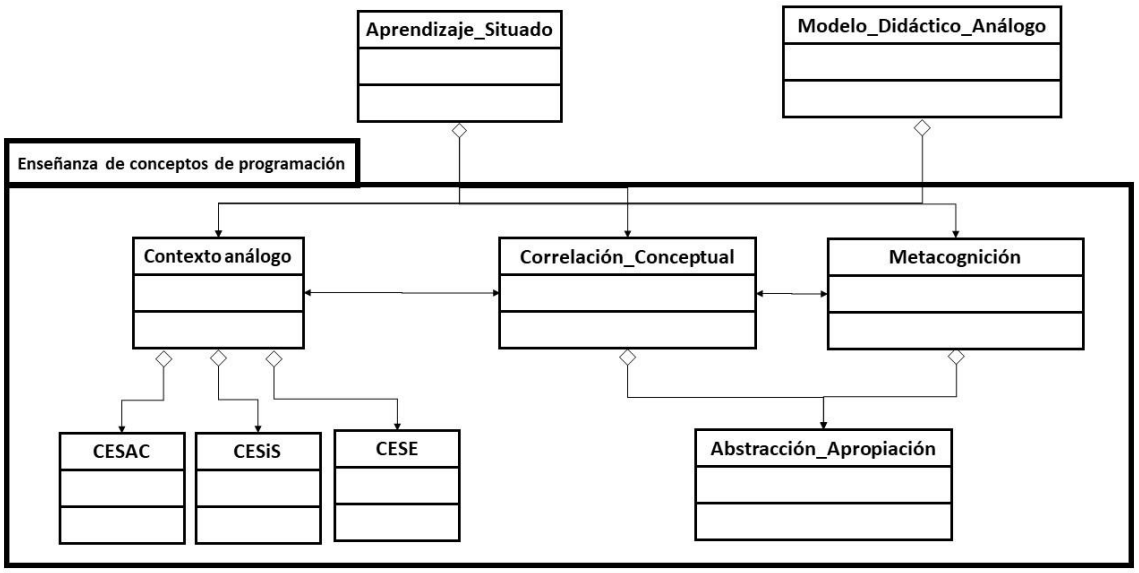


Figura 37. Modelo para la enseñanza de conceptos de programación
Fuente: elaboración propia

Además, en la Figura 22 del capítulo 3 se presentaron los elementos que debe tener una analogía para la enseñanza de conceptos fundamentales de programación de computadores que permitan el desarrollo de pensamiento algorítmico en un CS1.

4.2 Prácticas académicas en el entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico

En esta sección, se presenta a nivel de ejemplo, en primer lugar, cómo se llevó a cabo una sesión de clase con CodES y en segundo lugar cómo se realizó con el entorno de enseñanza propuesto.

4.2.1 CodES: Caso práctico

A continuación, se presenta uno de los ejemplos planteados en clase para la utilización de CodES en la enseñanza de los conceptos de Entrada y Salida. Este caso práctico presenta los artefactos por cada uno de los elementos del proceso de enseñanza establecidos en la Figura 26, que incluye la fase de: recolección de requerimientos,

análisis, diseño, codificación y prueba.

Enunciado del ejemplo: Capturar la base y la altura de un triángulo e imprimir su área.

Fase de Requerimientos. Una vez el estudiante identifique las palabras claves del enunciado del ejemplo, este construirá el formato de recolección de requerimientos como lo muestra la Tabla 31.

Formato de recolección de requerimientos	
Nombre del sistema	Cálculo área del triángulo
Requerimientos de entrada	Capturar base
	Capturar altura
Requerimientos de salida	Imprimir área

Tabla 31. Requerimientos área del triángulo
Fuente: elaboración propia

Fase de Análisis. Esta fase se apoya en dos artefactos: BlackBox y Graphic Interface Sketch (GIS). Es de aclarar que con la utilización de CodES, solo es necesario la construcción del BlackBox ya que la misma herramienta se encargará de automatizar los artefactos de las fases de análisis (GIS), diseño (Flow Chart) y codificación (Pseudocode).

El estudiante diseñará el BlackBox correspondiente al enunciado, teniendo como insumo el formato de recolección de requerimientos de la Tabla 6. Antes de diseñar el BlackBox en CodES se recomienda realizar el diagrama de Entrada-Salida (Figura 38) en el cuaderno del apuntes del estudiante, con el propósito de realizar procesos de abstracción y al mismo tiempo tener información para efectos de estudio posterior.

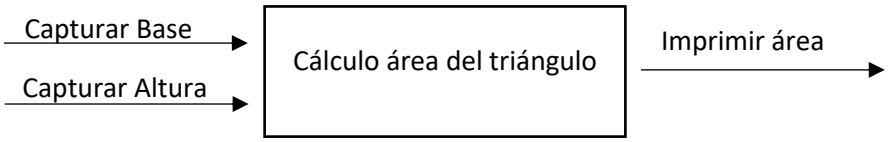


Figura 38. Diagrama de entrada-salida para cálculo de triángulo
Fuente: elaboración propia

Una vez construido el diagrama de Entrada-Salida, el estudiante utiliza CodES para su automatización, para ello, se recomienda que inicialmente se grafique el proceso central, para ello se utiliza la herramienta “Process” que al colocarlo en el entorno de CodES, este le solicita digitar el nombre para dicho proceso como lo muestra la Figura 39

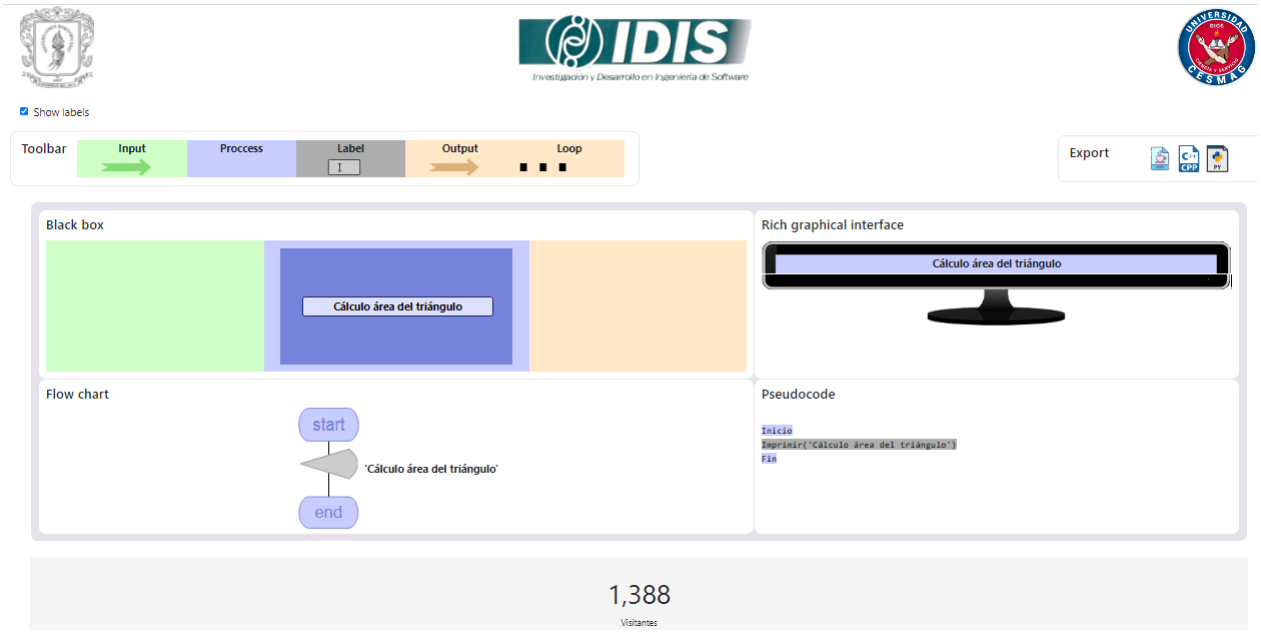


Figura 39. Inserción de proceso central en BlackBox
Fuente: elaboración propia

CodES por cada carácter ingresado en el título del proceso central, construye la Interfaz Gráfica Enriquecida (RGI) que le permite visualizar la apariencia computacional del ejercicio que se está construyendo. Al mismo tiempo se visualiza su componente del diagrama de flujo en el área de Flow Chart, y a la vez, su correspondiente Pseudocódigo como se exhibe en la figura 38.

Luego de ubicar el proceso central en el área de BlackBox en CodES, se debe adicionar una entrada utilizando la herramienta “Input”, la cual requiere la configuración de dos elementos: una variable (ubicada en el centro de la flecha) y un rótulo (que se presenta como rectángulo de línea punteada en la parte superior de la flecha). Dicho rótulo es necesario colocarlo con la herramienta “Label” y el propósito de este es indicarle al estudiante que los rótulos son elementos esenciales en la Interfaz de usuario, ya que estos guiarán adecuadamente al usuario final en el momento de interactuar con la aplicación. Dicha implementación de la entrada se muestra la figura 40.

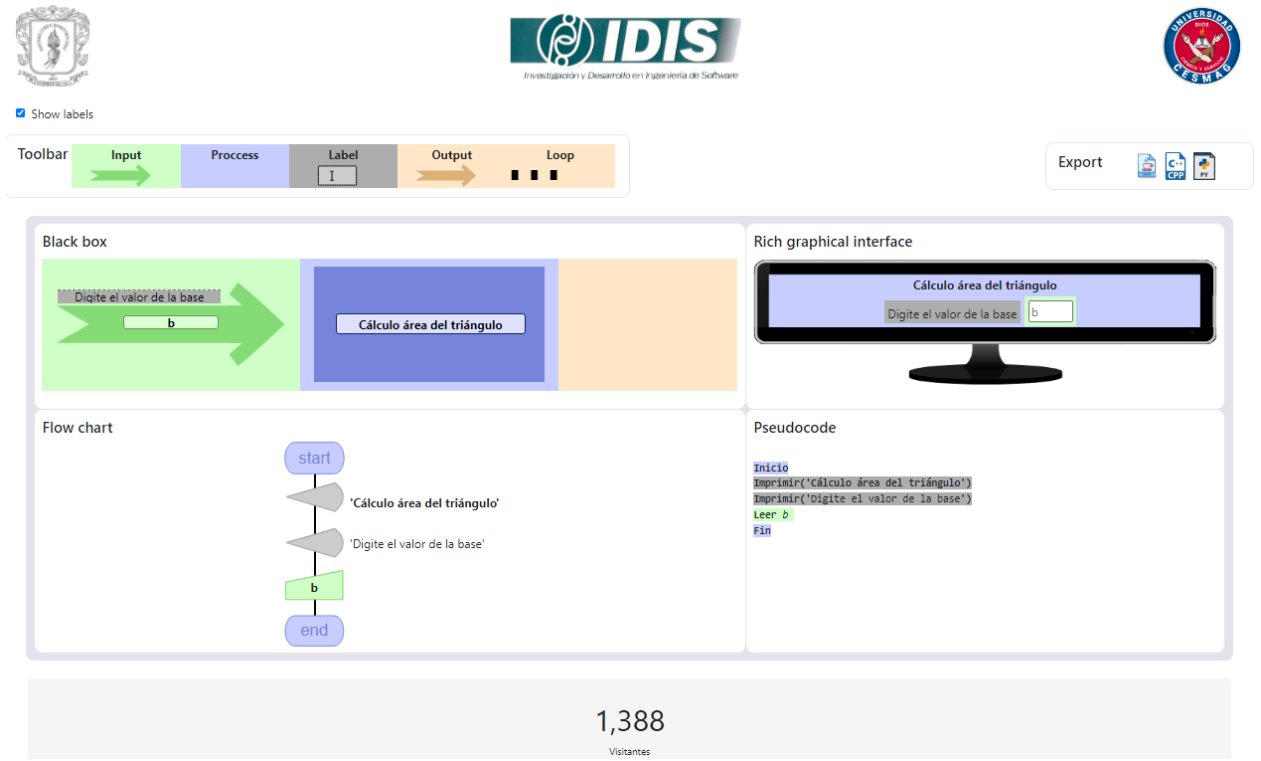


Figura 40. Inserción de entrada "base" en BlackBox

Fuente: elaboración propia

De igual manera en la Figura 39 se puede apreciar los artefactos de la fase de Análisis (BlackBox y GIS), Diseño (Flow Chart) y Codificación (Pseudocode) que se actualizan por cada modificación sobre la entrada configurada en CodES.

Para este ejemplo, es necesario volver a realizar el mismo proceso anterior para registrar la segunda entrada relacionada con la altura. Una vez ingresada y registrada su correspondiente variable y rótulo, su implementación en CodES se muestra en la Figura 41.

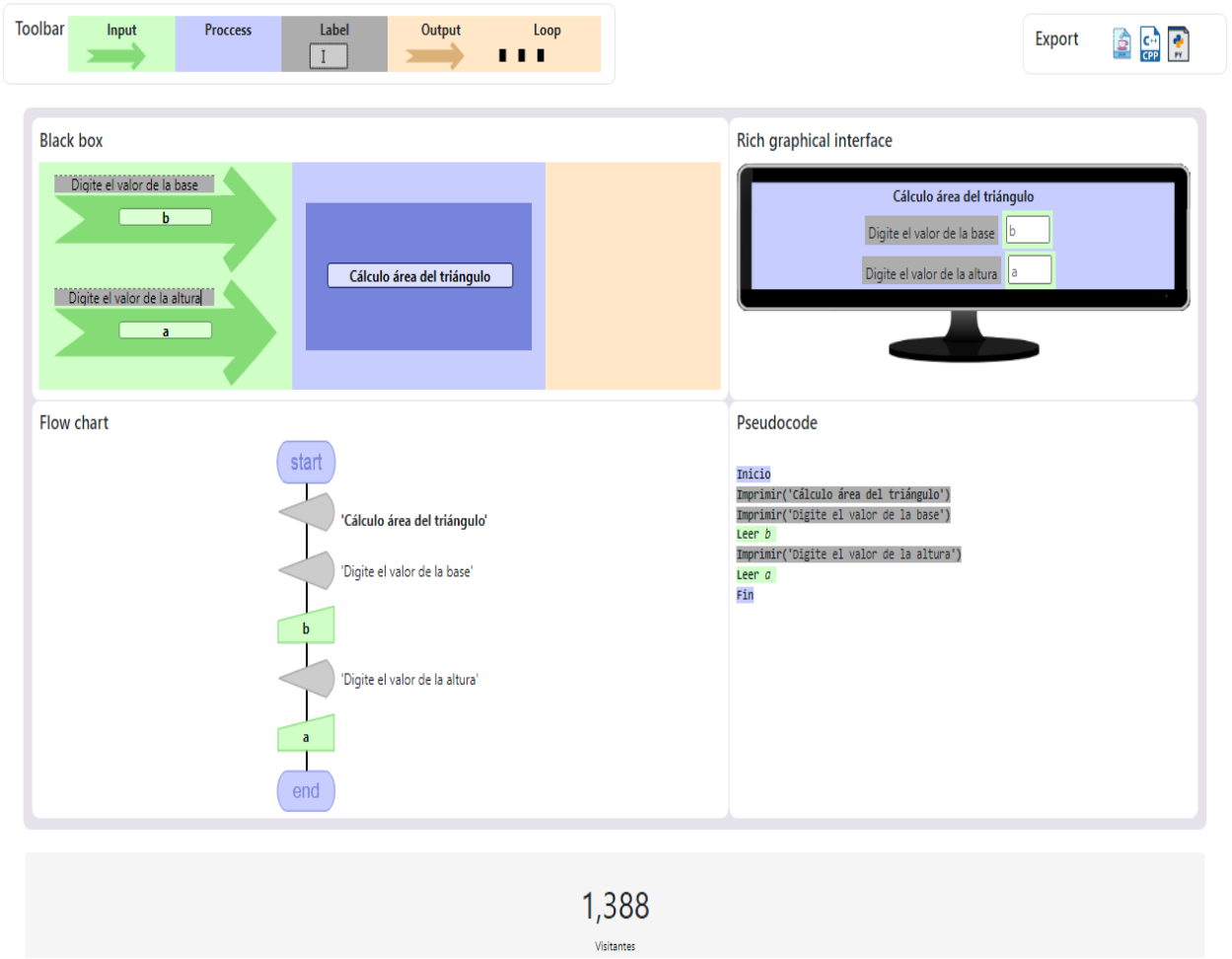


Figura 41. Inserción de entrada "altura" en BlackBox

Fuente: elaboración propia

Una vez registradas las entradas en el BlackBox, se ubica la salida a través de la herramienta "Output", que al igual que la herramienta "Input" requiere la configuración de dos elementos: Una operación que puede ser aritmética (la cual no requiere de una operación de asignación y en este caso será b*a/2) y un rótulo explicativo para dicha salida (que se realiza con la herramienta "Label").

CodES realiza la validación de las variables existentes en las entradas, de tal manera que, si estas no existen, se mostrará el recuadro en color rojo donde está la salida. Además, frente a este error, no se complementará ninguna acción en los artefactos de GIS, Flow Chart y Pseudocode. En el momento de configurar dicha salida, CodES exhibe automáticamente cada cambio en los artefactos antes mencionados. En la Figura 42 se presenta la implementación de la salida en CodES.

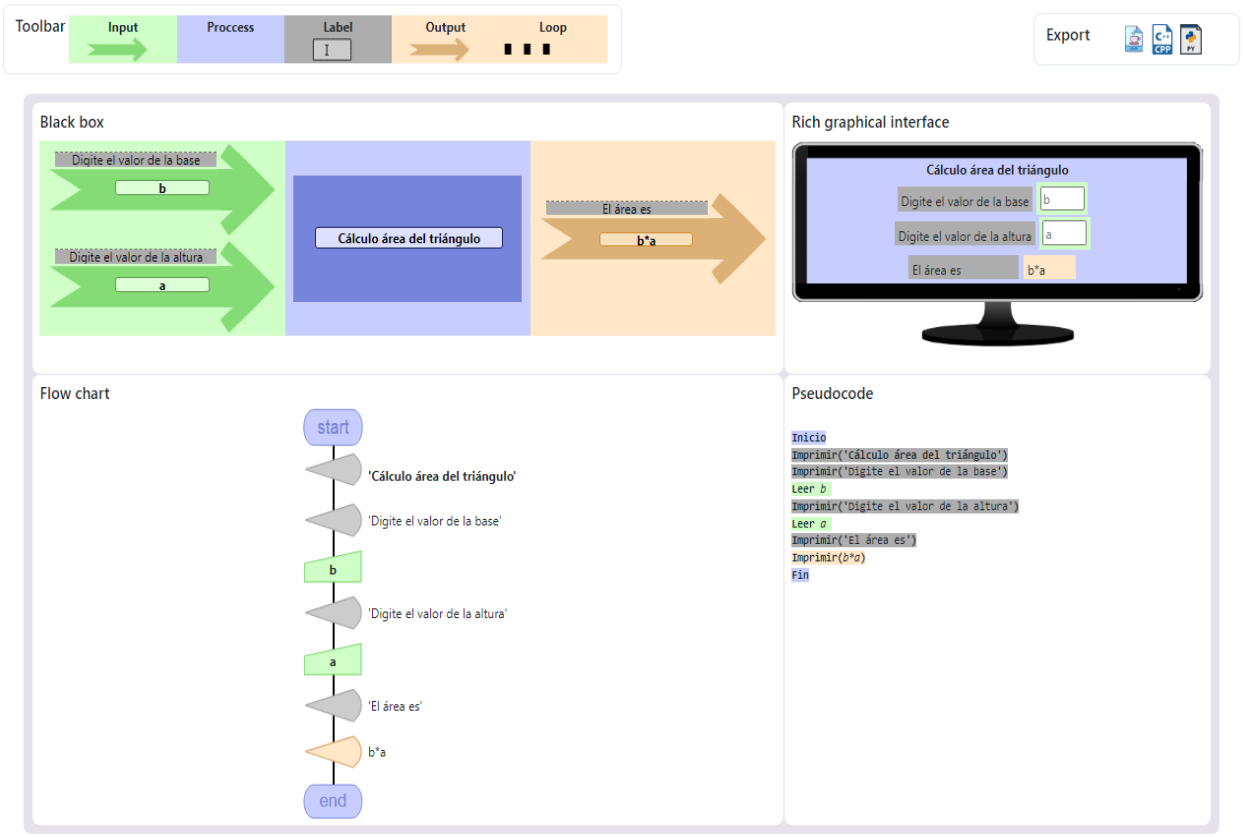


Figura 42. Inserción de Salida en BlackBox

Fuente: elaboración propia

Una vez terminado de construir el BlackBox, y para realizar la fase de Prueba, el estudiante puede ingresar valores en el artefacto para Rich Graphical Interface y comprobar el funcionamiento de la aplicación construida. CodES también valida las entradas numéricas en las correspondientes cajas de entrada, de tal manera que, si no se ha digitado un valor numérico, la entrada quedará vacía hasta que se digite correctamente dicho valor. En la figura 43 se presenta la ejecución numérica del ejemplo aquí planteado.

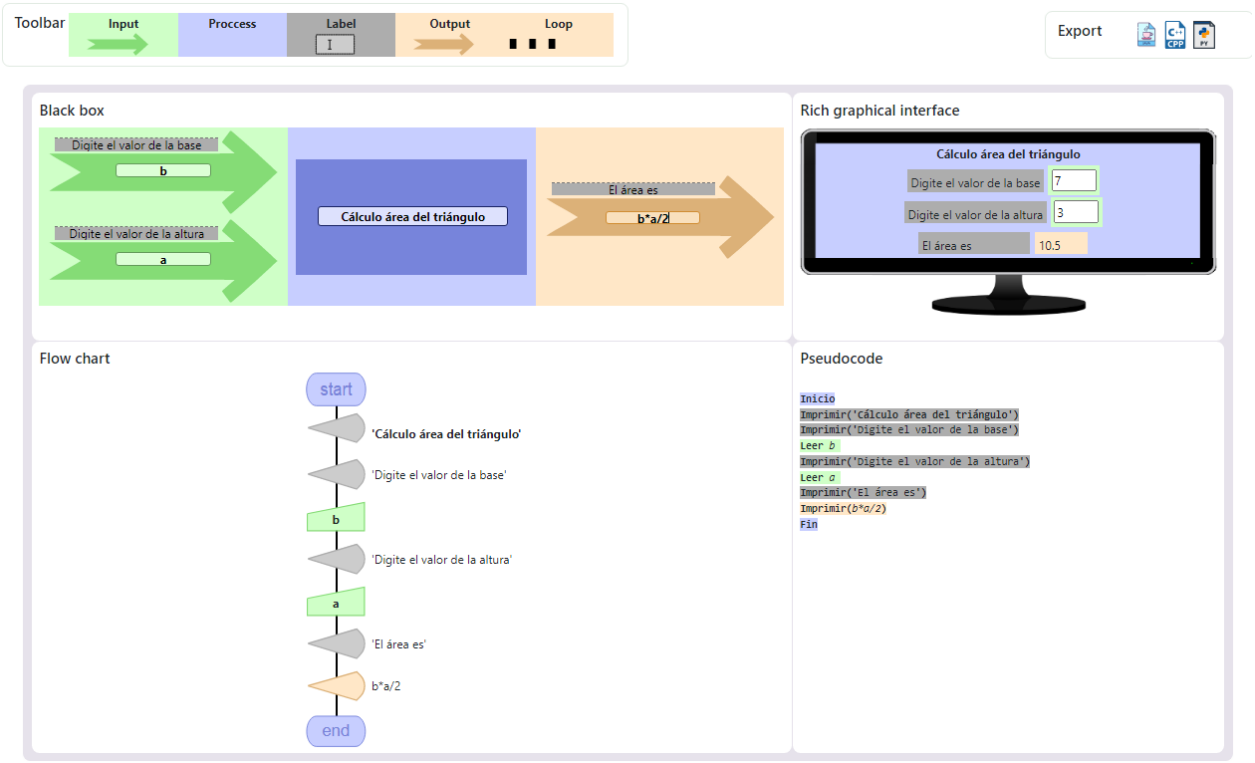


Figura 43. Prueba numérica en CodES

Fuente: elaboración propia

Finalmente, en desde la figura 39 hasta la Figura 43 se puede apreciar el desempeño de la impresora estática que utiliza CodES como guía en la visualización, esto es, utiliza un código de colores para cada uno de las herramientas en cada artefacto, así por ejemplo, cuando se coloca un objeto Input en el BlackBox, esta es una flecha de color verde y el mismo color se presenta en el TextBox correspondiente del Rich Graphical Interfaz, en el símbolo de entrada del Flow Chart y en el comando “Leer” del Pseudocode. Lo mismo con diferentes colores para las demás herramientas.

4.2.2 Sesión de clase con el entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico

Se tomó como ejemplo la sesión de clase para estudiar el concepto fundamental de Entrada, para ello, el profesor teniendo en cuenta el entorno presentado en la Figura 4 realizó los pasos indicados en el diagrama de actividad que se muestra en la Figura 44.

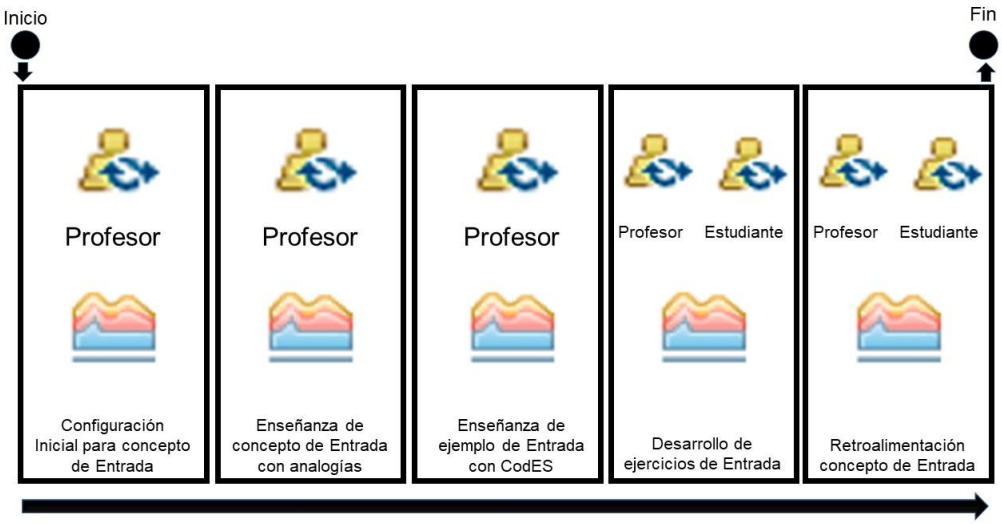


Figura 44. Diagrama de actividad de la clase para el concepto de Entrada
Fuente: elaboración propia

Para la enseñanza del concepto de Entrada presentado en la Figura 44 se realizaron las siguientes actividades:

- **Configuración Inicial para el concepto de Entrada**. En esta actividad el profesor teniendo en cuenta las recomendaciones presentadas en el modelo de descubrimiento para la construcción de analogías, el desarrollo de ejemplos computacionales con CodES y los ejercicios que se plantean para la apropiación del concepto de Entrada, realiza en el artefacto (Anexo 2) establecido el registro de dichos elementos de planeación.
- **Enseñanza de concepto de Entrada con analogías**. En esta parte, el profesor presenta a los estudiantes el concepto de Entrada a través de una analogía utilizando el formato sintáctico establecido en este estudio. A continuación, se hace un listad de algunas analogías que fueron utilizadas: “Digitar una clave en un cajero automático”, “Ingresar un alimento al microondas”, “Registrar un producto en la caja registradora”, “Entrar a un bus”.
- **Enseñanza del ejemplo de Entrada con CodES.** Una vez que el profesor presenta el concepto de Entrada mediante la analogía (en la que se utilizaron términos claves), el profesor presenta a los estudiantes un ejemplo de Entrada que puede ser modelado computacionalmente, para ello únicamente les presenta la instrucción de lo que es un diagrama de

entrada/salida y los invita a diseñar el ejemplo planteado mediante ese diagrama, el cual se hace inicialmente en el cuaderno de apuntes de cada estudiante. Uno de los ejemplos presentados en este estudio fue “Ingresar un número para calcular su cuadrado”. El profesor diseñó en el tablero el formato de requerimientos, con el cual se pasó a construir el Diagrama de Entrada/Salida el cual se implementó en CodES y permitió mostrar a los estudiantes paso a paso cómo CodES generaba automáticamente su Interfaz gráfica enriquecida de usuario (que permite ver una guía de cómo puede quedar la interfaz de usuario de la aplicación), además, los elementos que constituyen su diagrama de flujo y finalmente el Pseudocódigo generado.

- **Desarrollo de ejercicios de entrada.** Una vez realizado el primer ejemplo en CodES, el profesor plantea una serie de ejercicios para que los estudiantes los realicen y de esta manera afianzar el concepto de Entrada. Algunos de los ejercicios planteados fueron: “Capturar el valor de un lado en un triángulo para obtener su área”, “Digitar el valor de un producto y la cantidad de unidades a comprar para luego generar el valor de la compra”, “Ingresar el valor de la base y de la altura de un triángulo para calcular su área”.

Asimismo, en la Figura 45 se presenta el diagrama de actividad de detalle para el profesor del entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico con estudiantes universitarios en un CS1 para el concepto de “Entrada”.

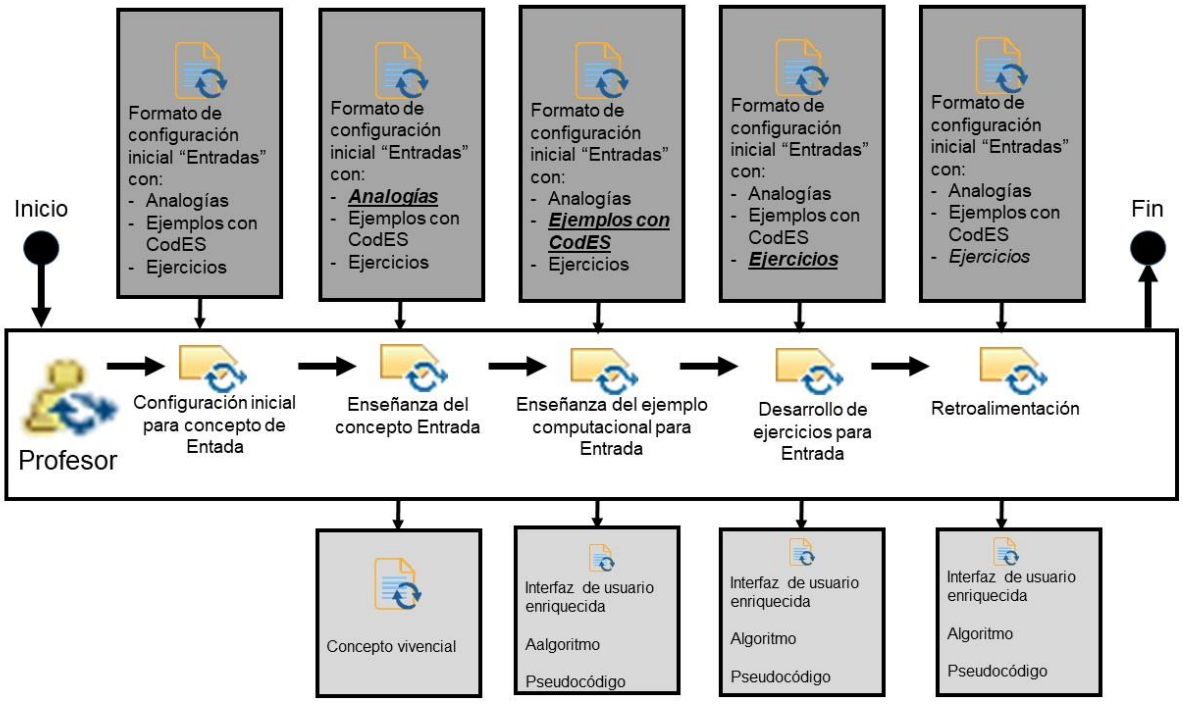


Figura 45. Diagrama de actividad de detalle para el profesor del entorno de enseñanza para "Entrada"
Fuente: elaboración propia

4.3 Evaluaciones

A continuación, se presentan las evaluaciones realizadas tanto al modelo de construcción de analogías, con el propósito de validar únicamente el modelo análogo construido en esta investigación, como también al entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios, que incluyó a dicho modelo análogo, la herramienta de visualización y demás elementos contemplados en el entorno.

4.3.1 Evaluación del modelo de construcción de analogías

El proceso de evaluación se realizó en dos momentos, en primer lugar, con profesores de un CS1 y luego con los estudiantes de dichos profesores. Para el primer momento de evaluación, se utilizó el paradigma positivista, con enfoque cuantitativo, utilizando el método empírico analítico, con un tipo de investigación descriptivo y mediante un pre-experimento con pre prueba y post prueba.

En este proceso experimental participaron dos profesores de dos universidades en Colombia: Universidad CESMAG (privada) y Universidad de Nariño (pública) del programa profesional denominado Ingeniería de Sistemas en los cursos de Introducción a la Programación y Fundamentos de Programación respectivamente, que se encuentra en el primer semestre académico de cada plan de estudios.

En consecuencia, el diseño experimental planteado fue G O_1 X O_2, donde G fue el grupo experimental conformado por los dos profesores que orientan los CS1 en cada universidad. Asimismo, O_1 fue conformado por el banco de ejemplos de analogías que a dichos profesores se les pidió que redactaran para explicar los conceptos de Entrada, Salida, Condicional y Ciclos, presentando un total de 14 ejemplos para el curso Introducción a la Programación y 17 ejemplos para Fundamentos de Programación.

Luego, a los profesores se le suministró el tratamiento experimental X que consistió en el modelo de creación de analogías propuesto en esta investigación. Finalmente, se les pidió que construyeran nuevamente las analogías por cada concepto, de acuerdo a las recomendaciones del tratamiento experimental y cuyo resultado fue O_2, obteniendo el mismo número de ejemplos iniciales planteados por cada profesor.

En un segundo momento, se evaluó la implementación de las analogías de acuerdo al banco de ejemplos obtenidas en O_2 con estudiantes de un CS1 y se comparó con los cursos anteriores orientados por los mismos profesores en los que utilizaron las analogías de O_1. Es este caso, el experimento tuvo el mismo paradigma, enfoque, método y tipo de investigación ya mencionados con los profesores, con la diferencia que el diseño fue experimental con grupos de control y experimentales con pre y post prueba.

En el proceso experimental participaron 154 estudiantes de las dos universidades ya mencionadas de la ciudad de San Juan de Pasto (Colombia) distribuidos en 4 grupos y cuyo diseño experimental para Universidad CESMAG fue: G_1 O_3 X O_4 y G_2 O_5 – O_6 y para Universidad de Nariño: G_3 O_7 X O_8 y G_4 O_9 – O_{10}.

Los grupos G_1 y G_3 corresponden a los grupos experimentales de cada universidad, mientras que G_2 y G_4 fueron sus grupos de control respectivamente, además, X fue el tratamiento experimental que consistió utilizar analogías con el modelo propuesto en esta investigación como estrategia didáctica de enseñanza de los conceptos fundamentales de programación para el desarrollo de pensamiento algorítmico en un CS1. A su vez, O_3, O_5, O_7, y O_9 fueron las pre pruebas mediante las cuales se evaluaron los conocimientos previos de los estudiantes antes de afrontar el CS1. Además, O_4, O_6, O_8, y O_{10} fueron las post pruebas realizadas al final del tratamiento experimental tanto para los grupos experimentales como los de control.

El grupo G_1 fue conformado por 45 estudiantes (89% hombres y 11% mujeres con edades entre 16 y 20 años, y 2 estudiantes repiten curso) del curso Introducción a la Programación de primer semestre de Ingeniería de Sistemas de la Universidad CESMAG, realizado entre febrero y mayo del 2019, a quienes se les aplicó O_3 mediante un cuestionario para verificar sus conocimientos previos en los conceptos de Entrada, Salida, Condicional y Ciclo. Luego, durante el curso se explicaron las temáticas mediante el tratamiento experimental X y finalmente las notas obtenidas en el proceso de evaluación académica fueron consideradas como pospruebas O_4.

Asimismo, el grupo de control G_2 estuvo conformado por 31 estudiantes (87% hombres y 13% mujeres, con edades entre 17 y 22 años y un estudiante repite el curso) que recibieron el mismo curso con el mismo profesor durante los meses de agosto y noviembre de 2018 en la misma universidad. A este grupo se le realizó el mismo cuestionario inicial que al grupo G_1 como pre prueba (O_5) pero no se les aplicó tratamiento experimental (x), es decir, el profesor utilizaba las analogías propuestas en O_1 y las notas obtenidas fueron consideradas como O_6.

Por otro lado, G_3 fue conformado por 40 estudiantes (83% hombres y 17% mujeres, con edades entre 16 y 25 años y ninguno repite el curso) del curso Fundamentos de Programación de primer semestre de Ingeniería de Sistemas de la Universidad de Nariño, realizado entre octubre 2018 y febrero de 2019, a quienes también se les aplicó O_7 como resultado de los conocimientos previos, luego, durante el curso se explicaron las temáticas mediante el tratamiento experimental X y finalmente las notas obtenidas en el proceso de evaluación académica fueron O_8.

De la misma forma, G_4 fue su grupo de control con 42 estudiantes (89% hombres y 11% mujeres, con edades entre 16 y 23 años, y ninguno repite el curso) que recibieron el mismo curso con el mismo profesor durante los meses de mayo y septiembre de 2018 en la misma universidad. A este grupo también se le realizó el mismo cuestionario inicial que al grupo G_3 como pre prueba (O_9) pero no se les aplicó tratamiento experimental (x) y las notas obtenidas fueron O_{10}.

El diseño evaluativo para los dos grupos experimentales y los dos de control se hizo mediante dos actividades de seguimiento académico: Un taller grupal (45%) y un quiz individual (55%) por cada uno de los temas propuestos en este estudio. Los promedios de las notas obtenidas en la pre y posprueba para los cuatro grupos se muestran en la Tabla 32.

Grupo	Tema	Pre prueba	Pos prueba
G_1	Entrada/salida	2,30	4,32
	Condicionales	1,70	3,56
	Ciclos	2,12	3,87
G_2	Entrada/Salida	2,43	4,03
	Condicionales	1,81	3,12
	Ciclos	2,40	3,26
G_3	Entrada/salida	2,6	4,67
	Condicionales	2,40	4,26
	Ciclos	2,91	4,39
G_4	Entrada/salida	2,57	4,28
	Condicionales	2,13	3,75
	Ciclos	2,12	4,00

Tabla 32. Promedio de notas de los grupos de control y experimentales.
Fuente: elaboración propia

4.3.2 Discusión de resultados para el modelo de construcción de analogías

Al realizar el análisis de las analogías planteadas en O_1 frente a obtenidas en O_2 se puede evidenciar que los profesores consideraron elementos como la estructura sintáctica, el contexto y el entorno análogo. Además, el primer profesor reemplazó 6 de los 14 ejemplos propuestos inicialmente y el segundo profesor cambió 5 de los 17 ejemplos. Asimismo, se pudo observar que el 100% de las analogías iniciales fueron reestructuradas de acuerdo a la recomendación sintáctica de utilizar un verbo, un sustantivo y un contexto, incorporando el uso de las palabras claves como verbo principal en cada tema enseñado.

Igualmente, todas las analogías en O_2 fueron más cortas en su longitud que las de O_1. Por ejemplo, la analogía para la enseñanza de condicionales compuestos presentada en O_1 cuya redacción fue “al preguntar por la edad de un estudiante usted debe establecer si corresponde a una persona mayor o menor de edad” en O_2 fue redactada así: “Evaluar

la edad de un estudiante y determinar si es mayor o menor de edad” siendo un 37,2% menor en extensión, manteniendo el contexto y adaptando la estructura sintáctica.

Asimismo, se aprecia que los profesores emplean analogías para diseño computacional, es decir, que pueden utilizarse para que el estudiante asimile el concepto a estudiar y que al mismo tiempo puedan modelarse computacionalmente.

También, en la modificación de las analogías por parte de los profesores se incluyeron los siguientes verbos para el concepto de Entrada: ingresar, escribir y digitar; para el concepto de Salida: Imprimir y calcular; para el concepto de Condicional: elegir, evaluar y comparar; y para el concepto de Ciclo: hacer, repetir, mientras y sumar.

De igual manera, los ejemplos presentados inicialmente por los profesores y los que se ajustaron después utilizan los siguientes contextos: para concepto de Entradas: recetas de cocina, operaciones aritméticas, ejercicios de notas de estudiantes y compras. Para el concepto de Salida: receta de cocina, nóminas y operaciones aritméticas y compras. Para el concepto de Condicional: menú, evaluación aritméticas, situaciones de evaluación de la vida estudiantil y algunas tareas decisiones del accionar cotidiano. Para el concepto de Ciclo: contextos aritméticos, cálculos y acciones escolares.

En el proceso de capacitación realizado con los dos profesores expertos acerca del manejo de analogías como una estrategia de enseñanza, se les informó del manejo de los tres entornos contemplados: Contexto de enseñanza situado en el aula de clase (CESAC), Contexto de enseñanza situado y simbólico (CESiS) y el Contexto de enseñanza situado y encubierto (CESE), siendo el SiCTE el de mayor utilización al presentar las analogías a los estudiantes.

Asimismo, se capacitó a los dos profesores en los dos tipos de analogías propuestos en este estudio Escenarios analógicos y analogías simples, siendo estas últimas las únicas utilizadas por los dos profesores.

Se puede apreciar también que, de acuerdo a las notas obtenidas por los estudiantes, el concepto de ciclo es el de menor valor cuantitativo en los promedios obtenidos, siendo quizá este concepto el que debe ser mayor atendido por el profesor en los próximos cursos CS1 para lograr un mejor acercamiento al estudiante de dicho concepto.

Por su parte, los conceptos de Entrada y Salida son los que mejores notas promedio obtienen, concluyendo que su proceso de aprendizaje es el de mayor facilidad para el estudiante al momento de su asimilación y estudio.

Las notas obtenidas en las pre pruebas tanto de grupos experimentales como de control nunca fueron superiores a las obtenidas en las diferentes pospruebas, concluyendo que los estudiantes tanto de la Universidad pública como de la privada

aprenden los conceptos fundamentales de la programación en un CS1, por ello la importancia de este estudio. Además, es evidente la dificultad presentada en el concepto de condicionales, obteniendo mejores resultados en el manejo de ciclos y logrando una buena apropiación en el manejo de entrada y salida de datos.

Finalmente, se realiza un análisis estadístico para determinar mediante la distribución de probabilidad T de Student [226], la diferencia que existe entre las notas obtenidas por el grupo experimental y las del grupo de control en cada universidad. Así, tanto en G1 como en G3, los valor estadístico t (3,46 y 2,06) son mayores tanto al valor crítico de t de una cola (1,67 en ambos casos) como al valor crítico para dos colas (1,99 en ambos casos), además, el registro para una y dos colas de P fue inferior al 5% en cada caso, lo cual concluye que las notas obtenidas por G_1 y G_3 frente a las del G_2 y G_4 en cada temática son estadísticamente diferentes.

Por lo tanto, el análisis estadístico anterior demuestra la incidencia que tuvieron las analogías construidas bajo el modelo aquí planteado, puede disminuir la complejidad de los temas para un estudiante en un CS1.

4.3.3 Evaluación del entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico

Finalmente, se validó el entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico que integra la propuesta metodológica para la enseñanza del concepto fundamental de programación computacional con estudiantes universitarios en un CS1 y la propuesta metodológica para la enseñanza del ejemplo fundamental computacional mediante CodES. Además, se evaluó CodES con un test de usabilidad y una técnica de seguimiento ocular.

El entorno de enseñanza fue evaluado bajo el paradigma positivista, con enfoque cuantitativo, utilizando el método empírico analítico, con un tipo de investigación descriptivo y mediante un diseño experimental con grupos de control y experimentales con post prueba. En el proceso investigativo participaron 147 estudiantes de dos universidades de la ciudad de San Juan de Pasto (Col) distribuidos en 6 grupos. En la Tabla 33 se muestra el diseño experimental aplicado en cada universidad:

Universidad	Diseño Experimental
Universidad CESMAG - UniCesmag (Privada)	G_1 X O_1 G_2 - O_2
	G_3 X O_3 G_4 - O_4
Universidad de Nariño - UDENAR (Pública)	G_5 X O_5 G_6 - O_6

Tabla 33. Diseño experimental por universidad.
Fuente: elaboración propia

Los grupos G_1, G_3 y G_5 corresponden a los grupos experimentales de cada institución y G_2, G_4 y G_6 fueron sus grupos de control respectivamente, además X fue el tratamiento experimental que consistió en el entorno de enseñanza. A su vez, O_1, O_2, O_3, O_4, O_5 y O_6 fueron las post pruebas realizadas al final del tratamiento experimental tanto para los grupos experimentales como los de control. En la Tabla 34 se encuentra en detalle la información de los grupos participantes.

Universidad	Facultad	Programa	Curso	Semestre	Jornada
UniCesmag	Ingeniería	Ingeniería de Sistemas	Introducción a la programación	1	Diurna
	Ingeniería	Ingeniería de Sistemas	Introducción a la programación	1	Nocturna
UDENAR	Ingeniería	Ingeniería de Sistemas	Fundamentos de programación	1	Diurna

Tabla 34. Caracterización de los grupos participantes
Fuente: elaboración propia

El primer grupo experimental G_1 fue conformado por 33 estudiantes del curso Introducción a la programación de primer semestre de Ingeniería de Sistemas del segundo periodo académico del 2019 a los cuales se les suministró el tratamiento

experimental X y finalmente se le aplicó una prueba posterior O_1. El grupo de control G_2 estuvo conformado por 27 estudiantes del periodo académico I-2019 del mismo semestre y curso del grupo experimental a quienes no se les aplicó tratamiento experimental y las notas obtenidas fueron consideradas como O_2.

El segundo grupo experimental G_3 estuvo constituido por 11 estudiantes de la jornada diurna de la misma universidad, curso y semestre que G_1, correspondientes al periodo académico I-2020 a quienes también se les aplicó el mismo tratamiento X con su pos evaluación O_3 donde su grupo de control G_4 fueron los 14 estudiantes de la jornada nocturna del mismo periodo académico, universidad, semestre y curso del grupo experimental a quienes tampoco se les aplicó el entorno de enseñanza, considerando también las notas obtenidas como O_4.

El tercer grupo experimental G_5 estuvo constituido por 29 estudiantes del curso de Fundamentos de programación de primer semestre de Ingeniería de Sistemas correspondientes al periodo académico II-2019 a quienes también se les aplicó el mismo tratamiento X con su correspondiente evaluación O_5 y su grupo de control G_6 fueron los 33 estudiantes del periodo académico I-2019 del mismo semestre y curso del grupo experimental a quienes tampoco se les aplicó tratamiento experimental, considerando también las notas obtenidas como O_6. En la Tabla 35 se aprecia la caracterización de los estudiantes participantes.

Grupo	Tipo	Estudiantes	Hombres	Mujeres	Edad (años)	Repiten asignatura
G_1	Experimental	33	27	6	18 a 21	0
G_2	Control	27	23	4	16 a 22	2
G_3	Experimental	11	11	0	18 a 35	0
G_4	Control	14	14	0	22 a 40	0
G_5	Experimental	29	24	5	18 a 20	0
G_6	Control	33	30	3	18 a 19	0

Tabla 35. Caracterización para estudiantes de Introducción a la Programación
Fuente: elaboración propia

Los grupos de control G_2, G_4 y G_6 fueron orientados por el mismo profesor, utilizando la metodología de enseñanza tradicional, en la cual se expone la temática de estudio, se llevan a cabo una serie de ejemplos y ejercicios, para finalmente realizar evaluaciones.

El proceso investigativo para los tres grupos de control, consistió en la realización de cuatro actividades evaluativas por cada unidad de competencia por cada semestre, que consistieron en la aplicación de dos talleres grupales con la participación de dos estudiantes (40%) y dos seguimientos individuales (60%), con el propósito de establecer cuantitativamente el proceso de apropiación de cada una de las unidades mencionadas con la metodología tradicional de clase. Los resultados de posprueba aplicados para los cursos de los grupos de control se muestran en la Tabla 36.

Grupo	Unidad de competencia	Taller 1	Taller 2	Total Talleres	Examen 1	Examen 2	Total Examen	Definitiva
G_2	Entrada/salida	4,30	4,00	4,15	3,80	4,00	3,90	4,00
	Condicionales	3,30	3,20	3,25	3,10	3,00	3,05	3,13
	Ciclos	4,00	3,80	3,90	3,70	3,60	3,65	3,75
G_4	Entrada/Salida	4,10	4,00	4,05	3,50	3,50	3,50	3,72
	Condicionales	3,90	3,80	3,85	3,70	3,20	3,45	3,61
	Ciclos	4,00	3,50	3,75	3,60	3,30	3,45	3,57
G_6	Entrada/salida	4,70	4,40	4,55	4,30	4,20	4,25	4,37
	Condicionales	4,30	4,00	4,15	4,00	4,00	4,00	4,04
	Ciclos	4,20	4,30	4,30	4,10	4,20	4,15	4,21

Tabla 36. Promedio notas grupos control
Fuente: elaboración propia

Por otra parte, y teniendo en cuenta los procesos de planeación establecidos en el entorno de enseñanza, para los grupos experimentales se diseñó un plan de actividades para acompañar dicho proceso en el CS1 que incluyó: pruebas de habilidades y recomendaciones de aprendizaje para los estudiantes y recomendaciones didácticas para los profesores.

Por ello, a los estudiantes de los grupos experimentales G_1, G_3 y G_5 se les aplicó un test de aptitud lógico matemático y de compresión lectora (Anexo 3) antes de comenzar el tratamiento experimental, esto con la intención de determinar los indicios de las habilidades de pensamiento lógico, pensamiento matemático y al menos tener indicadores de pensamiento crítico mediante su capacidad de abstracción lectora que se recomiendan en este estudio, con el fin de poder obtener posiblemente mejores resultados en un CS1, dichos promedios de notas obtenidas se presentan en la Tabla 37.

Grupo Experimental	Aptitud Matemática	Aptitud Lógica	Comprensión Lectora
G1	2,4	2,6	2.2
G2	2,6	2,8	2,6
G3	3,7	3,9	3,7

Tabla 37. Promedio notas para aptitudes de los grupos experimentales
Fuente: elaboración propia

Además, acorde con las sugerencias establecidas en este estudio, para contribuir con los procesos de autoaprendizaje, para los tres grupos experimentales se planeó un conjunto de ejercicios que los estudiantes debían realizar en su tiempo independiente (trabajo en casa) después de cada sesión de clase. También, para reforzar los procesos de aprendizaje en el aula, en al menos una sesión de clase se utilizó la técnica de aprendizaje cooperativo JigSaw [105] debido a los resultados positivos reportados por la literatura científica en los últimos años, ya que convierte a cada estudiante en elemento esencial para la resolución de una situación.

La actividad colaborativo con JigSaw consistió en que los estudiantes asumieron dos roles: analista y codificador. El rol de analista consistió tanto de la extracción de los requerimientos funcionales (de los enunciados de los ejercicios planteados) y su registro en el formato correspondiente, como de la elaboración del diseño del diagrama de entrada salida en papel (cuaderno), mientras que el codificador se encargó de plasmar dicho diseño en CodES con el análisis de los resultados de la interfaz gráfica enriquecida, el diagrama de flujo y su pseudocódigo.

En la actividad colaborativa se organizó el salón de clase de tal manera que los estudiantes quedaron reunidos en grupos de dos y distribuidos en todo el recinto, donde el profesor fue el encargado únicamente de presentar los enunciados que se encontraban en el formato de planeación de ejercicios de la configuración inicial. Cada ejemplo fue realizado por los dos estudiantes, cada uno en su rol de analista y codificador. Para ello, una vez presentado el enunciado del ejercicio, cada estudiante desarrolló su actividad desde su rol y la compartió con su compañero del grupo inicial, luego cada uno se reunió con otro “par” de otro grupo para analizar cada propuesta y luego esos dos se reunieron con otros “pares” hasta que la mitad del grupo quedó reunido bajo un mismo rol y la otra mitad con el otro rol para obtener una solución al ejercicio. Finalmente, cada estudiante volvió a su grupo inicial para compartir la solución final con su compañero y entre los dos retroalimentaron los resultados.

Por otro lado, se recomendó a los estudiantes que para el CS1 utilizaran dos técnicas de estudio que por su facilidad son propicias para un CS1: la primera consistió en la repetición en casa de los ejemplos desarrollados en clase sin ver previamente la solución para luego compararlas y la segunda formar parte de uno de los grupos de

estudios liderados por los estudiantes más “aventajados” de la clase. Cabe resaltar, que el profesor era el encargado de generar motivación en cada sesión de clase través de diferentes técnicas como: el discurso motivacional, los incentivos académicos por la participación activa en clase, por la realización de algunos ejercicios tanto en clase como después de ella, por la realización de otros ejercicios adicionales a los planeados en clase, entre otras.

Por su parte, al profesor se le solicitó que, desde la primera sesión de clase en cada grupo experimental, presentara y recordara continuamente las ventajas de utilizar los dos métodos de estudio antes mencionados. Además, se le pidió que en cada clase realice al menos una de las estrategias motivacionales ya descritas, conformara grupos de estudio seleccionado del mismo curso a los estudiantes de mayor rendimiento académico, planeara al menos una sesión de clase con la técnica colaborativa JigSaw, realizara al menos dos sesiones de clase con la dinámica de las maratones de programación y que todos los ejercicios presentados en clase sean de aprendizaje intencionados para que el estudiante formalice su conocimiento con motivación.

Con todo el despliegue de los procesos de planeación ya mencionados y una vez aplicado el tratamiento investigativo en cada grupo experimental G_1, G_3 y G_5 utilizando el entorno de enseñanza, se procedió a la realización de las mismas actividades evaluativas realizadas en los grupos de control. Los resultados de posprueba aplicados para los tres cursos muestran en la Tabla 38.

Grupo	Unidad de competencia	Taller 1	Taller 2	Total Talleres	Examen 1	Examen 2	Total Examen	Definitiva
G_1	Entrada/salida	4,50	4,70	4,60	4,60	4,40	4,50	4,54
	Condicionales	4,10	4,00	4,05	3,90	3,80	3,85	3,93
	Ciclos	4,40	4,30	4,35	4,00	4,00	4,00	4,14
G_3	Entrada/salida	4,40	4,80	4,60	4,20	4,40	4,30	4,42
	Condicionales	4,60	4,60	4,60	4,20	4,40	4,30	4,42
	Ciclos	4,50	4,40	4,45	4,30	4,10	4,20	4,30
G_5	Entrada/salida	5,00	4,80	4,80	4,80	4,88	5,00	4,80
	Condicionales	4,90	4,30	4,60	4,45	4,63	4,90	4,30
	Ciclos	4,65	4,60	4,80	4,70	4,68	4,65	4,60

Tabla 38. Promedio notas grupos experimentales

Fuente: elaboración propia

Finalmente, se realizó un análisis estadístico para determinar mediante la distribución de probabilidad T de Student, la cual se utiliza para examinar la diferencia entre dos muestras independientes y pequeñas [226], la diferencia que existió entre las notas obtenidas por el grupo experimental y las del grupo de control.

En la Tabla 39 se muestra el análisis comparativo entre el grupo experimental G_1 y el de control G_2 respecto a las tres unidades de competencia planteadas en la investigación para el CS1 de Introducción a la programación.

Tema	Concepto	Grupo experimental	Grupo de control
Entrada/Salida	Media	4,49575758	4,02962963
	Varianza	0,73820227	0,67062678
	Observaciones	33	27
	Varianza agrupada	0,70790981	
	Diferencia hipotética de las medias	0	
	Grados de libertad	58	
	Estadístico t	2,13490917	
	P(T<=t) una cola	0,01850238	
	Valor crítico de t (una cola)	1,67155276	
	P(T<=t) dos colas	0,03700477	
	Valor crítico de t (dos colas)	2,00171748	
Condicionales	Media	3,87545455	3,1437037
	Varianza	0,57988182	1,09901652
	Observaciones	33	27
	Varianza agrupada	0,81259738	
	Diferencia hipotética de las medias	0	
	Grados de libertad	58	
	Estadístico t	3,1281585	
	P(T<=t) una cola	0,0013754	
	Valor crítico de t (una cola)	1,67155276	
	P(T<=t) dos colas	0,0027508	

Tema	Concepto	Grupo experimental	Grupo de control
	Valor crítico de t (dos colas)	2,00171748	
Ciclos	Media	4,12484848	3,66148148
	Varianza	0,55394451	1,0152208
	Observaciones	33	27
	Varianza agrupada	0,76072353	
	Diferencia hipotética de las medias	0	
	Grados de libertad	58	
	Estadístico t	2,04726897	

Tabla 39. T de Student para G_1 y G_2 curso Introducción a la Programación

Fuente: elaboración propia

En la Tabla 40 se aprecia el resultado de la aplicación del tratamiento experimental en cada una de las unidades de competencia establecidas, de tal forma que G_1 posee un valor estadístico t (2,13490917, 3,1281585 y 2,04726897) mayor tanto al valor crítico de t de una cola (1,67155276 en cada uno) como al valor crítico para dos colas (2,00171748 en cada uno), además, el registro para una y dos colas de P fue inferior al 5% en los tres casos, lo cual concluye que las notas obtenidas por G_1 frente a las del G_2 en cada temática son diferentes estadísticamente.

De igual manera, se aplicó T de Student para los grupos G_3 y G_4 (ver Tabla 40) también del curso de Introducción a la programación, donde G_3 obtuvo un valor estadístico t mayor tanto al valor crítico de t de una cola como al valor crítico para dos colas, así también, el registro para una y dos colas de P fue inferior al 5% en los tres casos, concluyendo que las notas conseguidas por G_3 con relación a las del G_4 son diferentes estadísticamente en cada temática.

Grupo	Tema	Valor estadístico de t	valor crítico de t de una cola	valor crítico para dos colas
G_3	Entrada/salida	2,31176389	1,71387153	2,06865761
	Ciclos	2,16226104		
	Condicionales	2,40061536		
G_5	Entrada/salida	2,10911757	1,67064886	2,00029782
	Ciclos	2,26839432		
	Condicionales	2,39107334		

Tabla 40. Resultados T de Student para G_3 y G_5 curso Introducción a la Programación

Fuente: elaboración propia

Por último, al aplicar T de Student en cada una de las unidades de competencia en G_5 y G_6 (ver Tabla 40) para el curso de Fundamentos de programación, el grupo G_5 obtuvo un valor estadístico t mayor tanto al valor crítico de t de una cola como al valor crítico para dos colas, de igual forma, el registro para una y dos colas de P fue inferior al 5% también en los tres casos, así mismo, se concluye que las notas alcanzadas por el grupo G_5 son diferentes estadísticamente a las obtenidas por G_6 en cada temática.

4.3.4 Discusión de resultados del entorno de enseñanza basado en analogías para el desarrollo de pensamiento algorítmico

Una vez terminado el proceso metodológico, a continuación, se presenta los hallazgos más sobresalientes para este estudio.

Los resultados de las notas obtenidas en los tres cursos para grupos de control, evidencian la dificultad en el estudio de los condicionales, un mejor resultado en el manejo de ciclos y una buena apropiación en el manejo de captura y salida de datos. De igual manera, se puede concluir que las actividades realizadas en los talleres grupales obtienen mejores calificaciones que las individuales. También, los registros de notas obtenidos por los estudiantes de la universidad pública son mayores que los de la universidad privada.

En lo relacionado a los test de aptitud aplicados a los grupos experimentales, se puede observar que la aptitud matemática y la comprensión lectora obtienen los menores resultados, mientras que las valoraciones para aptitud lógica son sutilmente superiores. Asimismo, las calificaciones promedios de los grupos de la universidad privada en los tres test, se encuentran por debajo de la nota mínima (3.0) exigida en algunas universidades para aprobar un curso, mientras que los promedios de las notas de los estudiantes de la universidad pública están por encima de esta calificación.

Por su parte, los resultados obtenidos por los grupos experimentales en los talleres (que son grupales y estuvieron conformados por 2 estudiantes) son superiores a los exámenes individuales, lo cual demuestra que el estudiante adquiere mejores resultados trabajando en grupo que haciéndolo individual, lo cual concluye que el proceso llevado con el conjunto de estudiantes incide de manera positiva en sus resultados académicos.

De igual manera, se puede observar que la unidad de competencia relacionada con “Condicionales” es la que menor valor cuantitativo tiene frente a las otras dos unidades de competencia en los tres grupos experimentales, lo indica que los procesos

lógicos que requieren evaluación de proposiciones tienen un cierto grado de dificultad para los estudiantes.

Por otro lado, se puede apreciar en los tres grupos experimentales que en la unidad de competencia relacionada con ciclos el estudiante se siente más a gusto ya que sus notas mejoran con relación a la unidad de condicionales.

Al igual, se puede concluir que en la unidad de competencia “Captura y salida de datos” hay una buena apropiación por parte del estudiante ya que es la que más valor cuantitativo tiene en cada grupo con relación a las otras unidades de competencia. Además, Los resultados de las notas están por encima de la escala cuantitativa de 4.0, lo cual demuestra que hay una apropiación buena por parte de los estudiantes en forma general a finalizar dichos cursos.

Con los resultados hasta el momento analizados tanto de los grupos de control como de los experimentales, se evidencia la influencia que tiene el proceso de selección de los estudiantes en la universidad pública, frente a los de la universidad privada cuya política de ingreso generalmente es abierta.

Con los resultados individuales de los grupos de control y experimental analizados, a continuación, se hace el paralelo entre los grupos de control y los experimentales para cada uno de los cursos de este estudio. En la Tabla 41 se exhibe los resultados finales obtenidos por el grupo de control y experimental.

Grupo	Unidad de competencia	Grupo de control			Grupo experimental		
		Total Talleres	Total Examen	Definitiva	Total Talleres	Total Examen	Definitiva
G_1	Entrada/Salida	4,15	3,90	4,00	4,60	4,50	4,54
G_2	Condicionales	3,25	3,05	3,13	4,05	3,85	3,93
	Ciclos	3,90	3,65	3,75	4,35	4,00	4,14
G_3	Entrada/Salida	4,05	3,50	3,72	4,60	4,30	4,42
G_4	Condicionales	3,85	3,45	3,61	4,60	4,30	4,42
	Ciclos	3,75	3,45	3,57	4,45	4,20	4,30
G_5	Entrada/Salida	4,55	4,25	4,37	4,80	5,00	4,80
G_6	Condicionales	4,15	4,00	4,04	4,60	4,90	4,30
	Ciclos	4,30	4,15	4,21	4,80	4,65	4,60

Tabla 41. Grupo de control vs grupo experimental de Introducción a la Programación
Fuente: elaboración propia

La Tabla 41 muestra la incidencia que tuvo el tratamiento experimental propuesto en esta investigación para el curso de Introducción a la Programación del programa académico de Ingeniería de Sistemas de la Universidad CESMAG. Los datos presentados evidencian que las notas obtenidas por los grupos experimentales G1 y G2, son mayores a las registradas por el grupo de control tanto en las actividades grupales (Talleres) como en las individuales (Exámenes).

Así mismo, se muestra la incidencia que tuvo también el tratamiento experimental para el curso de Fundamentos de Programación del programa de Ingeniería de Sistemas de la Universidad de Nariño. Al igual que en la UniCesmag, los datos obtenidos evidencian que los resultados del grupo experimental son mejores a los del grupo de control.

A pesar de las bajas notas obtenidas por los dos grupo experimentales G_1 y G_2 en los test de aptitud matemática, lógica y comprensión lectora como indicio de pensamiento crítico, las notas finales obtenidas por los mismos grupos indica que posiblemente las estrategias de enseñanza y aprendizaje utilizadas en un CS1 sean más importantes que los presaberes con que cuentan los estudiantes para lograr los objetivos propuestos en este curso inicial.

A su vez, el análisis estadístico presentado con T de Student demuestra la incidencia que tiene el entorno de enseñanza en esta investigación en los grupos experimentales frente a los grupos de control, estableciendo que el desarrollo de pensamiento algorítmico en un CS1 se puede obtener al incorporar dicho entorno como una estrategia didáctica para disminuir la complejidad de las temáticas y así obtener resultados académicos que benefician de una manera directa a los estudiantes.

4.3.5 Evaluación de usabilidad para CodES

CodES fue evaluado por estudiantes, profesores y profesionales de la computación con el propósito de testear su usabilidad, para ello se utilizaron dos momentos. En primer lugar, se realizó con estudiantes bajo un diseño pre experimental como se describe a continuación:

$$\mathbf{G_3\ X\ O_1}$$

Donde:

G_3: fue el mismo grupo experimental ya mencionado, conformado por los 11 estudiantes del curso Introducción a la Programación del primer semestre de Ingeniería de Sistemas jornada diurna de la Universidad CESMAG, correspondientes al periodo académico I-2020.

X: Tratamiento experimental realizado bajo la técnica de Eye Tracking

O_1: Resultados obtenidos al aplicar **X**

Esta evaluación de usabilidad fue realizada con el primer ejemplo de estudio de Entradas/Salidas con CodES, correspondiente al “cálculo del cuadrado de un número”. Para llevar a cabo esta actividad, inicialmente se les solicitó a los estudiantes que diligenciaran el formato de recolección de requerimientos y el diagrama de entrada/salida en cada uno de sus apuntes. Luego, se instaló la versión Demo del software denominado RealEye [227] en las 11 estaciones de trabajo (computadores) de cada estudiante, configurando el testeo para sitio Web con la Url de CodES y luego con cada uno de ellos se llevó a cabo la calibración tanto de la postura adecuada de la cabeza como de los puntos de seguimiento ocular que el software verifica inicialmente.

Una vez calibrado RealEye, se les pidió a los estudiantes que realizaran dicho ejemplo en CodES y cuyos resultados del testeo ocular pueden observarse en la Figura 46, en la que se muestra el correspondiente mapa de calor promedio generado.

Figura 46. Mapa de calor CodES.
Fuente: elaboración propia

Por otro lado, en un segundo momento CodES fue evaluado por 6 Ingenieros de Sistemas de los cuales 3 de ellos pertenecen al Centro de Desarrollo de Software de la Universidad CESMAG y 3 son profesores de CS1 en cada una de sus Universidades en la ciudad de Pasto-Colombia (Universidad CESMAG, Universidad Mariana y Universidad de Nariño), bajo el siguiente diseño experimental:

$$G_1 \; X \; O_1$$

$$G_2 \; X \; O_2$$

Donde:

G_1: Grupo experimental conformado por 3 Ingenieros de Sistemas del Centro de Desarrollo de Software de la Universidad CESMAG.

X: Tratamiento experimental que consistió en aplicar el Test Expert Review Checkpoint.
O_1: Posprueba correspondiente a los resultados obtenidos al aplicar el tratamiento experimental **X** por e grupo G1**.**
G_2: Grupo experimental conformado por 3 profesores que orientan el CS1.
O_2: Posprueba correspondiente a los resultados obtenidos al aplicar el tratamiento experimental **X** por e grupo G2**.**

Para el tramiento experimental **X**, se utilizó el método de Evaluación Heurística de Nielsen [228] para valorar la usabilidad de CodES. Dicho método es aplicado por expertos y determina si los elementos de la Interfaz de Usuario de una aplicación de software cumplen los principios de usabilidad determinados en una lista heurística, detectando problemas de usabilidad a un menor coste que con las técnicas que utilizan usuarios.

Debido a que CodES funciona Online y para iniciar el proceso de Evaluación Heurística, se implementó el Test denominado Expert Review Checkpoint de Travis [229] (Anexo 20), el cual contempla 9 categorías, tomadas en este caso como los principios de evaluación. Dicho Test contó con 247 preguntas que conformaron la lista de la heurísticas a evaluar. A continuación, se describen los principios y algunas heurísticas utilizadas:

Página Principal. Es necesario establecer que su contenido sea útil, sus área de navegación estén bien definidas, contenga gráficos significativos, sus opciones de navegación estén ordenadas de manera lógica y jerárquica, que los usuarios determinen su importancia a primera vista, que las opciones principales estén claramente establecidas, entre otras. Este principio contempló 20 heurísticas.

Orientación a tareas y funcionalidades del sitio. Es importante evidenciar que el sitio esté libre de información irrelevante, que se eviten registros innecesarios, se minimice el número de pantallas por tarea, se tenga un mínimo de desplazamientos y clic, se haga la actividad del usuario de forma sencilla, cuando hay varias tareas se muestre todos los pasos, el uso de metáforas de interfaz sea fácilmente comprensible para el usuario, el sitio sea dirigido a usuarios con poca experiencia, entre otros. Este principio estableció 44 heurísticas.

Navegación. Es fundamental que el sitio tenga adecuadas formas de moverse entre las secciones, que las opciones de navegación estén ordenadas, que el sistema de navegación sea amplio y poco profundo, que la estructura sea simple, que las

herramientas se encuentren en la parte superior, que las etiquetas describan con precisión la información, que la terminología y convenciones sean apropiadas, entre otras. Este principio evaluó 29 heurísticas.

Formularios e ingresos de datos. Es importante que en el sitio, los campos de entrada de datos contengan valores predeterminados cuando sea apropiado, que las etiquetas de campo expliquen claramente qué entradas se desea, que los cuadros de texto tengan la longitud adecuada, que los menús desplegables, los botones de radio y las casillas de verificación se utilicen coherentemente, que con las pantallas de entrada de datos el cursor se ubique donde se necesita la entrada, que los usuarios pueden completar tareas simples ingresando solo la información esencial, que las etiquetas están cerca de los campos de entrada de datos, entre otros. Este principio incluyó 23 heurísticas

Confianza y credibilidad. Es necesario determinar que el sitio tenga un contenido autorizado y confiable, que evidencie que exista un responsable idóneo detrás del sitio, que evite guardar información para efectos de marketing, que sea fácil ponerse en contacto con alguien para obtener ayuda, que esté libre de errores tipográficos y ortográficos, entre otros. Este principio contempló 13 heurísticas

Escritura y calidad de contenido. Es fundamental que el sitio tenga contenido atractivo y único, que el texto sea conciso, que no tenga texto narrativo, que los elementos más importantes se coloquen en la parte superior, que la información esté organizada jerárquicamente, que el contenido esté creado específicamente para la web, que las oraciones se escriben con voz activa, que los títulos, subtítulos y párrafos sean cortos, entre otros. Este principio estableció 23 heurísticas

Diseño páginas. Es oportuno revisar que el sitio tenga una a densidad de la pantalla adecuada para los usuarios, que el diseño ayude a centrar la atención de la actividad, que el sitio se pueda utilizar sin desplazarse horizontalmente, que las opciones en las que se puede hacer clic sean funcionales, que la funcionalidad de los botones y controles sea obvia por sus etiquetas o por su diseño, que las fuentes sea utilizadas de forma coherente, que la relación entre los controles y sus acciones sea obvia, que los iconos y gráficos sean intuitivos, que los componentes de la GUI se utilicen de forma adecuada, que las fuentes sean legibles, entre otros. Este principio evaluó 38 heurísticas

Búsquedas. Se debe revisar que en el sitio la búsqueda sea intuitiva, que los resultados de la búsqueda son claros y útiles, que el motor de búsqueda maneje las consultas vacías con elegancia, que el cuadro de búsqueda sea lo suficientemente largo para manejar las longitudes de consulta comunes, que las búsquedas abarquen todo el sitio web, que la interfaz de búsqueda esté ubicada donde los usuarios esperarán encontrarla (parte superior derecha de la página), que el cuadro de búsqueda y sus controles estén claramente etiquetados, entre otros. Este principio incluyó 20 heurísticas

Ayuda, retroalimentación y tolerancia a fallos. Es necesario establecer que en el sitio las preguntas frecuentes o la ayuda proporcionen estén disponibles, que sea fácil obtener ayuda, que las indicaciones sea breves, que el usuario no necesite consultar manuales de usuario u otra información externa para utilizar el sitio, que los mensajes de error contengan instrucciones claras, que los mensajes de error estén escritos en un tono no irrisorio, que las páginas se carguen rápidamente (5 segundos o menos), que el sitio proporcione retroalimentación inmediata sobre la entrada o las acciones del usuario, entre otras. Este principio estableció 37 heurísticas

En la Tabla 42 se presentan los valores de medición establecidos para cada una de las 247 heurísticas evaluadas.

Valor	Medición	Observación
-1	No cumple	El aspecto evaluado no cumple totalmente con el criterio de valoración
0	Cumple parcialmente	El aspecto evaluado cumple parcialmente con el criterio de valoración
1	Cumple	El aspecto evaluado cumple totalmente con el criterio de valoración

Tabla 42. Valores y medición de las Heurísticas para el Expert Review Checkpoint
Fuente: Travis [229]

El Expert Review Checkpoint fue evaluado por los seis profesionales antes mencionados y cuyos resultados promedios, de acuerdo al diseño expetimental, se presentan en la Tabla 43.

Principios	No. Preguntas	No. Respuestas	G_1		G_2	
			Puntos	Puntaje	Puntos	Puntaje
Página principal	20	20	16	20%	16	20%
Orientación a tareas y funcionalidades del sitio	44	44	28	44%	34	44%
Navegación	29	29	19	29%	21	29%
Formularios e ingresos de datos	23	23	20	23%	17	23%
Confianza y credibilidad	13	13	9	13%	12	13%
Escritura y calidad de contenido	23	22	13	23%	16	23%
Diseño páginas	38	38	31	38%	29	38%
Búsquedas	20	20	-17	20%	-4	20%
Ayuda, retroalimentación y tolerancia a fallos	37	37	21	37%	26	37%
Total	247	247	247	76,3%		83%

Tabla 43. Resultados promedios de Expert Review Checkpoint para G_1 y G_2
Fuente: elaboración propia

4.3.6 Discusión de resultados de usabilidad para CodES

Una vez aplicadas las pruebas de usabilidad por parte de estudiantes como de profesores y profesionales de la computación a CodES, fue posible evidenciar las siguientes situaciones:

Al indagar a los estudiantes del G_3 sobre el uso de técnicas de evaluación ocular, expresaron que el 100% grupo nunca las habían utilizado, demostrando una nueva motivación al elegir el programa académico de Ingeniería de Sistemas al observar la aplicabilidad de este tipo de herramientas en múltiples contextos.

Al mismo tiempo, al aplicar la prueba de testeo ocular al ejemplo realizado en CodES por los 11 estudiantes del G_3, se observó que el mapa de calor promedio indicó el interés de dicho grupo en primer lugar en la zona de trabajo de BlackBox, seguido del Rich Graphical Interface, luego del Flow Chart y finalmente del Pseudocódigo, como lo muestra la Figura 46. Es decir, cada vez que el estudiante realizaba una acción en el

BackBox, inmediatamente fijaba su atención en la forma en que dicho componente se presentaba en la interfaz gráfica (a través del Rich Graphical Interface), en seguida observó cuál era su representación como algoritmo (en el Flow Chart) y finalmente terminó revisando cómo se representaba en pseudocódigo (mediante el Psudocode).

De acuerdo a la forma en que el estudiante utilizó CodES en este ejemplo, fue la intención de este estudio, sustentado en que el desarrollo inicial de pensamiento algorítmico para constructores de software debe estar basado en los procesos genéricos del desarrollo de software que contemplan: los requerimientos (al diligenciar el formato de requeriientos propuesto), análisis (en la construcción del diagrama de entrada/salida), diseño (en la visualización de la interfaz gráfica enriquecida y el diagrama de fulo generado en Flow Chart) y codificación (en el algoritmo escrito en Pseudocódigo).

Por otro lado, de los resultados promedios al aplicar el test del usabilidad mediante Expert Review Checkpoint a G_1 y G_2 se pudo apreciar que los profesionales expertos en desarrollo de softtware calificaron a CodES con un 73,6% donde el principios de búsquedas fue factor más crítico en la asignación de puntos y el mejor evaluado con mayor cantidad de puntos correspondió a Formularios e ingresos de datos.

A su vez, los profesores del CS1 que conformaron el G_2 calificaron a CodES con un 83%, donde los principios de búsquedas también fueron los de menor asignación de puntos y el mejor evaluado correspondió a Confianza y credibilidad.

Los resultados anteriormente expuestos dan cuenta posiblemente que desde G_1 los profesionales en desarrollo de software realizaron una evaluación de CodES desde su funcionalidad, en cambio los profesores que conformaron G_2 lo hicieron adicionando el propósito didáctico de CodES.

Realizando el promedio de las evaluaciones hechas por G_1 y G_2, se obtuvo un valor de 79% en el puntaje total de CodES, que lo categoriza como un “buen resultado” y con elementos por mejorar como su sistema de búsquedas y el diseño gráfico de su Frontend.

A su vez, en la Tabla 44 se muestran algunas de las heurísticas categorizadas como “bien valoradas” de acuerdo con el nivel de evaluación del test de usabilidad aplicado a CodES mediante Expert Review Checkpoint.

No	Características “bien Valorada”
1	El aplicativo Web permite establecer que la información se presenta en un orden simple, natural y lógico
2	El sitio requiere un mínimo de desplazamiento y clic
3	Los usuarios pueden completar tareas rápidamente
4	El sitio está dirigido a personas con poca experiencia en la web
5	Los elementos de la página de inicio se centran claramente en las tareas clave de los usuarios
6	Las categorías de productos se proporcionan y son claramente visibles en la página de inicio

No	Características “bien Valorada”
7	Las áreas de navegación en la página de inicio no están sobre formateadas y los usuarios no las confundirán con anuncios
8	La página de inicio contiene gráficos significativos, no imágenes prediseñadas o imágenes de modelos
9	Existe una forma conveniente y obvia de moverse entre las páginas y secciones relacionadas y es fácil volver a la página de inicio
10	La estructura del sitio es simple, con un modelo conceptual claro y sin niveles innecesarios

Tabla 44. Resultados de Expert Review Checkpoint para característica “bien valorada”

Fuente: elaboración propia

De otra parte, en la Tabla 45 se muestran las heurísticas que “deben mejorase” de acuerdo a los resultados del test de usabilidad aplicado a CodES.

No	Características “Deben mejorarse”
1	La búsqueda predeterminada es intuitiva de configurar (sin operadores booleanos);
2	La página de resultados de búsqueda muestra al usuario lo que se buscó y es fácil de editar y volver a enviar la búsqueda
3	Los resultados de la búsqueda son claros, útiles y están clasificados por relevancia
4	La página de resultados de búsqueda deja en claro cuántos resultados se recuperaron y el usuario puede configurar el número de resultados por página
5	Si no se devuelven resultados, el sistema ofrece ideas u opciones para mejorar la consulta basándose en problemas identificables con la entrada del usuario
6	el motor de búsqueda maneja las consultas vacías con elegancia; el motor de búsqueda proporciona una revisión ortográfica automática y busca plurales y sinónimos
7	Cuando una tarea involucra documentos de origen (como un formulario en papel), la interfaz es compatible con las características del documento de origen
8	El sitio ingresa automáticamente datos de formato de campo; el mismo formulario se utiliza tanto para iniciar sesión como para registrarse
9	Los formularios advierten al usuario si se necesita información externa para completarlos, entre otros

Tabla 45. Resultados de Expert Review Checkpoint para característica “deben mejorarse”

Fuente: elaboración propia

Capítulo 5

Conclusiones, trabajos futuros y productos construidos

5.1 Conclusiones

De acuerdo con los resultados obtenidos en este estudio, al evaluar tanto la construcción de las analogías como el entorno de enseñanza propuesto, se puede apreciar que los logros alcanzados por los grupos experimentales frente a los conseguidos por los grupos de control, permiten afirmar que se cumple la hipótesis de investigación Hi propuesta, que establece que la utilización de un entorno de enseñanza basado en analogías es una estrategia que mejora el desarrollo de pensamiento algorítmico en un CS1 para estudiantes universitarios, con relación a otras estrategias utilizadas en estos cursos.

Una de las principales actividades para lograr el cumplimiento del primer objetivo específico, fue el desarrollo de una Revisión Sistemática de Literatura realizada en esta investigación, mediante la cual fue posible reafirmar que en primer lugar los reportes más citados en la literatura científica corresponden a los problemas existentes en la enseñanza del pensamiento algorítmico, el desconocimiento de las temáticas y la falta de asociación de los conceptos fundamentales de programación con la propia experiencia de los estudiantes. Además, en segundo lugar, se encuentran los inconvenientes relacionados con estilos de aprendizaje, las pocas semanas de estudio para aprender las temáticas requeridas en un CS1 y las frecuentes actualizaciones de los lenguajes de programación. Y finalmente en un tercer momento se caracterizan las estrategias de enseñanza aplicadas a un CS1, en las que se menciona la existencia de procesos basados en el Modelo Didáctico Analógico pero su documentación es mínima en la literatura científica. Igualmente, los autores en consenso manifiestan la importancia de

los algoritmos en los procesos iniciales de un CS1 y su utilidad no solo computacional sino en la solución de actividades cotidianas que incluyen creatividad y procesos abstractos. Por su parte, el desarrollo de pensamiento algorítmico en un CS1 es el elemento de mayor importancia y complejidad a la vez, puesto que, de los logros e inconvenientes presentados en este primer momento, se reflejarán los buenos o deficientes resultados que sin duda afectarán el desempeño y asimilación de los procesos computacionales requeridos en los cursos posteriores.

De igual manera, se encontró que a pesar de que la enseñanza de la algoritmia se ha fortalecido en los últimos tres decenios, aún no existe un consenso en las actuales universidades e instituciones de educación superior que forman constructores de software a nivel profesionales, en la forma cómo se debe afrontar un CS1 en lo relacionado con los mecanismos de instrucción, las herramientas, las metodologías, las didácticas, los saberes, las competencias, los modelos y demás elementos necesarios para lograr importantes resultados en este campo. Todo lo anterior, permitió realizar una caracterización de procesos y herramientas requeridas en el desarrollo de pensamiento algorítmico que fueron el insumo para desarrollar el segundo objetivo específico planteado en este estudio.

A la vez, para alcanzar el segundo objetivo específico, se realizaron cinco investigaciones exploratorias que permitieron reafirmar la importancia que tienen las analogías como estrategias de enseñanza para el desarrollo de pensamiento algorítmico. Teniendo en cuenta estos estudios exploratorios y los hallazgos reportados en el primer objetivo específico, se construye un entorno de enseñanza basado en analogías para un CS1 con estudiantes universitarios, que combina una propuesta metodológica para la enseñanza únicamente del concepto fundamental de programación a través de un modelo de descubrimiento de conocimiento, con una propuesta metodológica para la enseñanza del ejemplo fundamental computacional, complementada con una serie de ejercicios de repaso y un proceso de retroalimentación por parte del profesor. El diseño construido en este segundo objetivo específico fue el insumo para alcanzar el tercer objetivo planteado en esta investigación.

De igual manera, para lograr el tercer objetivo específico, primero se validó la propuesta metodológica para la enseñanza del concepto fundamental de programación para desarrollar pensamiento algorítmico, que fue obtenida a través de un modelo de descubrimiento de conocimiento, la cual fue validada con profesores de un CS1 y luego

con los estudiantes de dichos profesores. En un segundo momento se validó el entorno de enseñanza aquí propuesto incluyendo a CodES como facilitador del proceso. Los resultados obtenidos en el proceso de validación permitieron comprobar estadísticamente la incidencia del entorno en al menos los registros académicos logrados por los estudiantes de los grupos experimentales frente a los obtenidos por los grupos de control.

Asimismo, en esta investigación se recomienda que al afrontar un CS1 es necesario fortalecer los niveles de abstracción a través del desarrollo de pensamiento algorítmico y potenciarlo con herramientas de visualización y la utilización de un lenguaje de programación que minimice al máximo su sintaxis de codificación. Además, es necesario la realización de ejemplos inicialmente relacionados con experiencias ya adquiridas por los estudiantes como el caso de las situaciones planteadas en los entornos de la Física, Química, Matemática, Geometría, Estadística, entre otros y combinarlos con ejemplos cotidianos cercanos a las experiencias de los estudiantes, como por ejemplo, el sistema cuantitativo de notas, compras, facturación, etc., para luego llevarlos a formular proyectos pequeños con la adopción de algunas herramientas, metodologías y estrategias individuales y colectivas.

Además, un importante patrón encontrado en el modelo de descubrimiento de conocimiento realizado con el conjunto de las analogías recopiladas y que son utilizadas por los 33 profesores que participaron, determina que dichos profesores incluyen en la redacción de las analogías utilizadas para el desarrollo de pensamiento algorítmico en un CS1, palabras claves que son propias de la terminología para ejemplos computacionales. Asimismo, la mayoría de contextos utilizados en las analogías son de tipo cuantitativo, donde el campo numérico tiene especial importancia.

Igualmente, un importante hallazgo en la estructura de las oraciones utilizadas por los profesores al redactar sus propias analogías para la enseñanza de los conceptos fundamentales para desarrollar pensamiento algorítmico en un CS1, radicó en que el elemento gramatical de mayor valor establecido por las técnicas de NLP aplicadas fue el verbo, el cual se convirtió en el elemento importante de este estudio ya que fue asociado con la palabra clave para la asimilación del concepto a enseñar y que a la vez fue también el atributo principal en las tareas de clasificación, asociación, segmentación y es el elemento clave para el funcionamiento de CodES.

También, otro resultado importante del modelo de descubrimiento de conocimiento radica en que los ejemplos numéricos, las recetas de cocina y los juegos son las categorías más utilizadas como referente para la construcción de analogías que permitan al estudiante abstraer un concepto fundamental de programación de computadores.

En el mismo sentido, en la etapa de descubrimiento con la tarea de minería del modelo de conocimiento aquí presentado, se puedo evidenciar que los resultados obtenidos al analizar el banco de analogías compiladas mediante aprendizaje no supervisado y semi supervisado, afirmaron y complementaron los resultados conseguidos en aprendizaje supervisado, siendo los primeros más específicos y los segundos de mayor generalidad. Asimismo, las reglas de asociación generadas en este estudio presentaron un mayor nivel de confianza que las encontradas en las reglas de clasificación, además, permitieron obtener patrones de mayor especificidad por cada uno de los conceptos evaluados. A su vez, los resultados encontrados por agrupamiento ratificaron las reglas obtenidas por asociación y clasificación por extensión en los 4 conceptos fundamentales de programación.

Al mismo tiempo, los resultados del análisis lingüístico que complementaron el modelo de descubrimiento de conocimiento, permitieron observar que a pesar de tener una amplia variedad léxica en los 4 conceptos de los fundamentos de programación considerados en este estudio, y debido a la amplia zona geográfica de donde se tomaron los datos, los profesores utilizan analogías en contextos comunes.

Asimismo, el modelo de descubrimiento de conocimiento para la construcción de analogías combinó elementos morfosintácticos con los hallazgos más importantes resultantes como son: la inclusión de palabras claves propias de la terminología de ejemplos computacionales, contextos cuantitativos, verbos en infinitivo, sustantivos comunes y el análisis sentimental de las oraciones. Obteniéndose los elementos que se sugiere tenga una analogía para la enseñanza de dichos conceptos mediante la utilización de oraciones con sujeto elíptico o tácito las cuales no poseen en la estructura gramatical el sujeto explícito, es decir, tienen la forma de instrucciones y se categoriza en el modo gramatical imperativo afirmativo.

Con lo anterior, los hallazgos de la etapa de descubrimiento en la tarea de minería mediante el análisis de aprendizaje supervisado, no supervisado, semi supervisado, combinado con técnicas de análisis lingüístico, analítica textual y datos enlazados,

permitieron seleccionar los verbos, sustantivos y contextos adecuados para los conceptos de Entrada, Salida, Condicional y Ciclo presentados en este estudio con el propósito de acercar de mejor forma al estudiante a la abstracción de dichos conceptos para desarrollar pensamiento algorítmico.

Por otro lado, y teniendo en cuenta la propuesta metodológica para enseñar el ejemplo fundamental computacional, se puede establecer que la enseñanza de pensamiento algorítmico en potenciales programadores de computadores en un CS1 debe tomar como base los componentes estructurales del proceso de desarrollo de software para obtener resultados de aprendizaje evolutivos, ya que, desde este nivel, el estudiante debe comprender que la construcción de software obedece a una colección de actividades planificadas y bien diferenciadas.

Otro resultado importante de este estudio radica en que el desarrollo de pensamiento algorítmico es la clave fundamental en el aprendizaje de los diferentes tipos de paradigmas de programación y en un CS1 debe ser cuidadosamente planeado por el profesor, debido a que los resultados de aprendizaje logrados por el estudiante serán la base para afrontar esquemas de pensamiento de mayor complejidad en los siguientes niveles de aprendizaje. Además, el pensamiento matemático estable, el pensamiento lógico armonioso y el pensamiento crítico en formación, son las habilidades iniciales que preferiblemente un estudiante debe tener al iniciar un CS1 para lograr resultados de aprendizaje en un menor tiempo.

Por su parte, dentro del entorno de enseñanza planteado en este estudio, se desarrolló CodES como una herramienta de visualización que permite fortalecer la iniciación del pensamiento algorítmico en un CS1 mediante el artefacto más simple de análisis computacional que es el diagrama de entrada y salida, con el propósito de generar proceso de abstracción en el estudiante tanto para el diseño de algoritmos como para su escritura. CodES tiene un Frontend inspirado en un diseño simple y minimalista, actualmente es una aplicación inscrita ante la oficina de registro de la Dirección Nacional de Derecho de Autor en Colombia y se encuentra disponible a través de http://micodes.com.co/

Asimismo, es importante tener en cuenta que el entorno de enseñanza aquí propuesto, permite que el estudiante centre su atención en comprender en primer lugar el problema a solucionar a través de sus elementos esenciales y a la vez intuir desde un inicio la

interfaz computacional a construir y sus respectivos diagramas de diseño y codificación, utilizando palabras claves de identificación que le permitirán de una forma sencilla al estudiante el manejo de entradas y salidas e identificar cuando debe incorporar estructuras condicionales y repetitivas en sus algoritmos.

Es importante destacar que en este estudio no se pudo medir con amplitud las recomendaciones contempladas en el entorno de enseñanza propuesto, es decir, la incidencia de contemplar las habilidades que tiene un estudiante antes de afrontar un CS1 y las estrategias de aprendizaje sugeridas, así como las estrategias didácticas de enseñanza recomendadas para el profesor, sin embargo, en los test de aptitud aplicados a los grupos experimentales, se puede observar que la aptitud matemática y la comprensión lectora obtienen los menores resultados, mientras que las valoraciones para aptitud lógica son sutilmente superiores; pero a pesar de las bajas notas obtenidas por los grupo experimentales en dichos test, las notas finales logradas por los mismos grupos indica que posiblemente las estrategias propuestas en el entorno de enseñanza, sean más eficientes que los presaberes con que cuentan los estudiantes para lograr los objetivos propuestos en este curso inicial.

5.2 Trabajos futuros

Como trabajo futuro se pretende construir un software recomendador que permita evaluar una analogía propuesta por el profesor y determinar si es adecuada o no al concepto fundamental de un CS1 que se quiere enseñar y si además tiene los componentes sintácticos necesarios para su buena comprensión.

Así también, se pretende validar el entorno de enseñanza con estudiantes de educación primaria y bachillerato dada la facilidad de aprendizaje propuesta en dicho entorno y así poder establecer su incidencia en los procesos de aprendizaje escolar a temprana edad y en estudiantes adolescentes con el fin de realizar procesos de desarrollo de pensamiento algorítmico y validar posiblemente su incidencia en la solución de problemas diferentes a los computacionales.

Por otro lado, se puede ampliar este estudio teniendo en cuenta las recomendaciones relacionadas con las habilidades y estrategias de aprendizaje que debe tener un estudiante antes de iniciar un CS1 y las recomendaciones didácticas que debe cumplir

un profesor, evaluándolas y midiendo el nivel de correlación existente en los resultados obtenidos.

De igual manera se pretende complementar el entorno de enseñanza con estrategias de aprendizaje colaborativas y poder evaluar en principio cómo esta combinación puede mejorar los resultados ya obtenidos en este estudio. Además, sería conveniente también evaluar la misma experiencia con estudiantes de educación primaria y bachillerato para inicialmente poder desarrollar pensamiento algorítmico inicial y evaluar al mismo tiempo su incidencia como una herramienta que apoye la estructuración y solución de problemas en cualquier área de saber humano.

También, se proyecta adicionar en CodES nuevas funcionalidades para el manejo modular de instrucciones, el envío de valores y variables por parámetro y referencia, la posibilidad de realizar ejemplos de mayor tamaño y complejidad computacional y a la vez complementar las opciones de exportación a código fuente en determinados lenguajes de programación.

Por otro lado, existe la posibilidad de extender este entorno a la enseñanza de la programación avanzada, es decir, para abordar temáticas como recursividad, teoría de colas, listas, árboles, etc. y al mismo tiempo construir una nueva herramienta computacional basada en software visualizador que permita automatizar los ejemplos computacionales de una forma ágil y sencilla.

Sería importante realizar un estudio que permita establecer las posibilidades de éxito o fracaso que puede tener un estudiante antes de iniciar un CS1, con el propósito de determinar sus fortalezas y debilidades para potenciarlas antes o durante el estudio del mismo curso.

5.3 Productos construidos

Como resultado del proceso investigativo llevado a cabo en este estudio, se han construido productos de investigación basados en artículos científicos (ver Tabla 46), software educativo (ver Tabla 47), participación en eventos científicos (ver Tabla 48), libros de investigación (ver Tabla 49), proyectos de trabajo de pregrado (ver Tabla 50), maestría (ver Tabla 51) y doctorado (ver Tabla 52).

No	Título	Indexación	Autores	Detalle
1	Discovery Model Based on Analogies for Teaching Computer Programming	**Revista: Mathematics** **Q1 en JCR** **A1 en Publindex**	Javier Jiménez César Collazos Manuel Ortega	https://doi.org/10.3390/math9121354
2	Considerations for the Teaching-Learning Processes of Introductory Programming Courses: A Systematic Literature Review	**Revista: Tecnológicas** **B en Publindex**	Javier Jiménez César Collazos Óscar Revelo	https://doi.org/10.22430/22565337.1520
3	Collaborative work as a didactic strategy for teaching/learning programming: a systematic literature review	**Revista: Tecnológicas** **B en Publindex**	Óscar Revelo César Collazos Javier Jiménez	https://doi.org/10.22430/22565337.731
4	CodES: Visualization tool for developing algorithmic thinking	**Revista: Campus Virtuales** **Q3 en JCR** **B en Publindex**	Javier Jiménez César Collazos Manuel Ortega	Aceptado para publicación enero. junio 2022
5	Algorithmic thinking and extension of its definition for training software developers: a	**Revista: IEEE Rita** **Q3 en JCR** **A2 en Publindex**	Javier Jiménez César Collazos Manuel Ortega	En proceso de evaluación desde mayo de 2021-

No	Título	Indexación	Autores	Detalle
	systematic literature mapping			

Tabla 46. Artículos científicos

No	Título	Descripción	Registro
1	CodES: Codificación con Entradas y Salidas	Software educativo para el desarrollo de pensamiento algorítmico	En proceso en la Dirección Nacional de Derecho de Autor en Colombia

Tabla 47. Software educativo

No	Título	Evento	Publicación	Autores	Detalle
1	User Interface Sketch for the Development of Algorithmic Thinking with Augmented Reality	Proceedings of the XIX International Conference on Human Computer Interaction	**ACM (2018)**	Javier Jiménez César Collazos Manuel Ortega Miguel Redondo	https://doi.org/10.1145/3233824.3233843
2	Collaborative Strategy with Augmented Reality for the Development of Algorithmic Thinking	Proceeding Human-Computer Interaction 4th Iberoamerican Workshop, HCI-Collab 2018	**Springer. Communications in Computer and Information Science CCIS 2018**	Javier Jiménez César Collazos Manuel Ortega Miguel Redondo	https://doi.org/10.1007/978-3-030-05270-6_6

Tabla 48. Eventos científicos

No	Título	Autores	Isbn
1	La metacognición y la teoría de la actividad en la enseñanza de la programación (Anexo 12)	Romero Chaves, Olga Cristina Rosero Sosa, María Mercedes Jiménez Toledo, Javier Alejandro	978-958-56354-4-9
2	Propuesta metodológica para el desarrollo del pensamiento computacional en la formación de Contadores Públicos desde el componente informático mediante hojas de cálculo (Anexo 13)	Deixy Ximena Ramos Rivadeneira Javier Alejandro Jiménez Toledo	978-958-5504-59-2

Tabla 49. Libros resultados de investigación

No	Título	Estudiantes	Programa	Estado
1	ALGORITBUILD: aplicativo para potenciar el desarrollo del pensamiento algorítmico en estudiantes de primer curso de programación (Anexo 14)	Erika Daniela Reyes Cuarán Saidy Fernanda Tumal Guzmán	Universidad CESMAG Ingeniería de Sistemas	Terminado
2	Plataforma para el desarrollo de pensamiento algorítmico (Anexo 15)	Wilmer Andrés Menes Botina	Universidad CESMAG Ingeniería de Sistemas	Terminado
3	Realidad virtual para el desarrollo de pensamiento algorítmico en profesores universitarios: AlgoVirtual	Daniel Esteban Madroñero Muñoz	Universidad CESMAG Ingeniería de Sistemas	Terminado

No	Título	Estudiantes	Programa	Estado
	(Anexo 16)	Christian Daniel Goyes Muñoz		

Tabla 50. Proyectos de pregrado

No	Título	Estudiantes	Programa	Estado
1	Técnica de conformación de grupos en escenarios de aprendizaje colaborativo basado en rasgos de la personalidad para la enseñanza de la programación	Óscar Revelo Sánchez (Anexo 17)	Maestría en Computación Universidad del Cauca	Terminada
2	Modelo de descubrimiento para la gestión de estudiantes en escuelas de conducción automovilísticas	Jhon Jairo Rivera Minayo	Maestría en Ingeniería de Sistemas y Computación Universidad de Nariño	En curso
3	Modelo basado en HCI para la construcción de Websites universitarios para personas invidentes	Víctor Alejandro Paredes Solarte	Maestría en Ingeniería de Sistemas y Computación Universidad de Nariño	En curso

Tabla 51. Proyectos de Maestría

No	Título	Estudiantes	Programa	Estado
1	Modelo colaborativo para la generación automática de mapas de irradiancia a través de imágenes satelitales multiespectrales	Mag. Héctor Mora (Anexo 18)	Doctorado en Ciencias de la Electrónica Universidad del Cauca	Inicia febrero 2022
2	Diseño e implementación de un Robot social terapéutico para disminuir el impacto	Mag. Joan Ayala	Doctorado en Ciencias de la Electrónica	Inicia febrero 2022

No	Título	Estudiantes	Programa	Estado
	cognitivo en pacientes diagnosticados con Alzheimer desde un enfoque de Diseño Centrado en el Usuario	(Anexo 19)	Universidad del Cauca	

Tabla 52. Proyectos de Doctorado

Bibliografía

[1] J. A. Jiménez Toledo, C. A. Collazos, and M. Ortega, “Discovery model based on analogies for teaching computer programming,” *Mathematics*, vol. 9, no. 12, pp. 1–21, 2021, doi: 10.3390/math9121354.

[2] J. López Reguera, C. Hernández Rivas, and Y. Farran Leiva, “An automatic evaluation platform with an effective methodology for teaching / learning in computer programming,” *Ingeniare. Rev. Chil. Ing.*, vol. 19, no. 2, pp. 265–277, 2011, doi: 10.4067/S0718-33052011000200011.

[3] B. Depetris, D. A. Mallea, H. Pendenti, G. Tejero, and G. Prisching, “Teaching and Learning Programming and Concurrent Programming with DaVinci,” in *X Congress of Technology in Education and Education in Technology*, 2015, pp. 194–202, [Online]. Available: http://sedici.unlp.edu.ar/handle/10915/48587.

[4] G. Silva, P. Arjona, and F. Castillo, “More Time or Better Tools? A Large-Scale Retrospective Comparison of Pedagogical Approaches to Teach Programming,” *IEEE Trans. Educ.*, vol. 59, no. 4, pp. 274–281, 2016, doi: 10.1109/TE.2016.2535207.

[5] J. Insuasti, “Problemas de enseñanza y aprendizaje de los fundamentos de programación * Problems of teaching and learning the basics of programming Problemas de ensino e aprendizagem dos fundamentos de programação,” *Rev. Educ. y Desarro. Soc.*, vol. 10, no. 2, pp. 12011–5318, 2016, doi: 10.18359/reds.1701.

[6] J. A. Jiménez-Toledo, C. Collazos, and O. Revelo-Sánchez, “Considerations in the teaching-learning processes for a first course in computer programming: a systematic review of the literature,” *TecnoLógicas*, vol. 22, pp. 83–117, Dec. 2019, doi: 10.22430/22565337.1520.

[7] R. Muñoz, M. Barría, R. Noel, E. Providel, and P. Quiroz, “Determinando las dificultades en el aprendizaje de la primera asignatura de programación en estudiantes de ingeniería civil informática,” in *XVII Congreso Internacional de Informática Educativa*, 2012, pp. 120–126.

[8] A. Carbone, J. Hurst, I. Mitchell, and D. Gunstone, “An exploration of internal factors influencing student learning of programming.,” *Proc. Elev. Australas. Conf. Comput. Educ.*, vol. 95, pp. 25–34., 2009, [Online]. Available: https://www.academia.edu/17067234/An_exploration_of_internal_factors_influenci

ng_student_learning_of_programming.

[9] R. . Felder and R. Brent, “Understanding Student Differences,” *J. Eng. Educ.*, vol. 94, no. 1, pp. 57–72, 2005, doi: https://doi.org/10.1002/j.2168-9830.2005.tb00829.x.

[10] S. Casas and V. Vanoli, “Programación y Algoritmos: Análisis y Evaluación de Cursos Introductorios,” 2007.

[11] M. Ortega *et al.*, “IProg: Development of immersive systems for the learning of programming,” *ACM Int. Conf. Proceeding Ser.*, vol. Part F1311, p. 6, 2017, doi: 10.1145/3123818.3123874.

[12] C. Watson and F. Li, “Failure rates in introductory programming revisited,” in *Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE ’14*, 2014, pp. 39–44, doi: 10.1145/2591708.2591749.

[13] NSF, “Science and engineering indicators 2012,” *National Sciience Foundation*, Arlington, VA, USA., 2012.

[14] W. Dann, S. Copper, and R. Pausch, *Learning to program with Alice. Upper Saddle River, NJ.* 2006.

[15] A. Saez, C. Febe, U. Puentes, and J. Menéndez, “El desarrollo de la habilidad : implementar algoritmos . Teoría para su operacionalización The development of the skill : implement algorithms . Theory for its implementation,” *Rev. Cuba. Ciencias Informáticas*, vol. 9, no. 3, pp. 99–112, 2015, [Online]. Available: https://rcci.uci.cu/?journal=rcci&page=article&op=view&path%5B%5D=1108&path%5B%5D=351.

[16] M. Bozorgmanesh, M. Sadighi, and M. Nazarpour, “Increase the eficiency of adult education with the proper use of learnin g styles,” *Nat. Sci.*, vol. 9, no. 1, p. 5, 2011, [Online]. Available: http://www.sciencepub.net/nature/ns0905/21_5408ns0905_140_145.pdf.

[17] C. Malliarakis, M. Satratzemi, and S. Xinogalos, “Educational games for teaching computer programming,” *Res. e-learning ICT Educ. Technol. Pedagog. ical Instr. Perspect. Springer*, vol. 1, no. June, pp. 99–118, 2014, doi: 10.1007/978-1-4614-6501-0.

[18] L. P. Baldwin and J. Kuljis, “Learning programming using program visualization techniques.,” 2001, [Online]. Available: http://www.computer.org/portal/.

[19] J. R. Olague Sánchez, S. Torres Ovalle, F. Morales Rodríguez, A. G. Valdez Menchaca, and A. E. Silva Ávila, “Sistemas De Gestión De Contenidos De Aprendizaje Y Técnicas De Minería De Datos Para La Enseñanza De Ciencias Computacionales,” *Investigacion*, vol. 15, no. 45, pp. 391–421, 2012, [Online]. Available: http://www.redalyc.org/pdf/140/14012507004.pdf.

[20] E. Dunican, “Making the analogy: Alternative delivery techniques for first year programming courses. In J. Kuljis, L. Baldwin & R. Scoble (Eds.),” in *Proceedings from the 14th Workshop of the Psychology of Programming Interest Group, Brunel*

University, 2002, pp. 89–99, [Online]. Available: http://www.ppig.org/sites/default/files/2002-PPIG-14th-dunican.pdf.

[21] B. Depetris, “Experiencias con Da Vinci Concurrente en la enseñanza inicial de la programación y la programación concurrente,” 2013, [Online]. Available: http://hdl.handle.net/10915/27581.

[22] L. Thomas, M. Ratcliffe, J. Woodbury, and E. Jarman, “Learning styles and performance in the introductory programming sequence,” in *Proceedings of 33rd SIGCSE Technical Symposium*, 2002, vol. 34, pp. 33–37, doi: https://doi.org/10.1145/563340.563352.

[23] I. Tamouli, E. Doyle, and M. Huggard, “Establishing structured support for programming students,” 2004.

[24] W. Hartman, J. Nievergelt, and R. Reichert, “Kara, finite state machines, and the case for programming as part of general education.,” 2001.

[25] V. Frittelli *et al.*, “Desarrollo de Juegos como Estrategia Didáctica en la Enseñanza de la Programación,” Córdoba, Argentina, 2013. [Online]. Available: http://conaiisi.frc.utn.edu.ar/PDFsParaPublicar/1/schedConfs/4/120-429-1-DR.pdf.

[26] J. Sánchez García, M. Urías Ruiz, and B. Gutiérrez Herrera, “Análisis de los problemas de aprendizaje de la Programación Orientada a Objetos,” *Ra Ximha*, vol. 11, no. 4, pp. 289–304, 2015, [Online]. Available: https://drive.google.com/file/d/0B_QQ0W8TI5acTWRwOWZWLV9FWE0/view.

[27] L. Spigariol and N. Passerini, “Enseñando a programar en la orientación a objetos,” in *Congreso Nacional de Ingeniería Informática / Sistemas de Información. Educación en Ingeniería.*, 2013, vol. 1, [Online]. Available: http://conaiisi.frc.utn.edu.ar/PDFsParaPublicar/1/schedConfs/4/97-498-1-DR.pdf.

[28] R. López, “Metodología para el desarrollo de la lógica de la programación orientada a objetos,” *Sist. Cibernética e Informática*, vol. 10, no. 2, pp. 27–329, 2013, [Online]. Available: http://www.iiisci.org/journal/risci/FullText.asp?var=&id=CA889XD13.

[29] E. Lahtinen, K. Ala-Mutka, and H. Jarvinen, “A Study of Difficulties of Novice Programmers,” *Innov. Technol. Comput. Sci. Educ.*, vol. 1, pp. 14–18, 2005, doi: 10.1145/1067445.1067453.

[30] M. A. Sicilia, “Más allá de los contenidos: compartiendo el diseño de los recursos educativos abiertos,” *Univ. Knowl. Soc. J.*, vol. 4, no. 1, pp. 1698–580, 2007, doi: http://dx.doi.org/10.7238/rusc.v4i1.297.

[31] G. M. Rodríguez Carrillo, “Enseñanza de la programación de computadoras para principiantes: un contexto histórico,” *INVENTUM*, vol. 9, no. 17, pp. 51–61, Jul. 2014, doi: 10.26620/uniminuto.inventum.9.17.2014.51-61.

[32] O. I. Buriticá Trejos, “Modelo de enseñanza con aprendizaje colaborativo en estudiantes de programación de computadores,” *Vector*, vol. 9, pp. 48–58, 2014.

[33] C. Chaves and M. M. Rosero, “Teaching model and its relationship with

metacognitive processes in systems programming," *Rev. Educ. en Ing.*, vol. 9, no. 17, pp. 1–12, 2014, doi: https://doi.org/10.26507/rei.v9n17.334.

[34] J. J. A. Pimentel, O. S. Nieva García, R. Solar Gonzales, and G. Arista López, "Software para la enseñanza-aprendizaje de algoritmos estructurados," *Rev. Iberoam. Educ. en Tecnol. y Tecnol. en Educ.*, vol. 8, pp. 23–33, 2012, [Online]. Available: http://teyet-revista.info.unlp.edu.ar/TEyET/article/view/253.

[35] V. Sarienko, "Didactic function of forming algorithmic thinking," *Prof. docente Asp. teóricos y Metod.*, vol. 8, pp. 91–99, 2018, doi: 10.31865/2414-9292.8(1).2018.153742.

[36] E. Semenihina and J. Rudenko, "Problems of learning programming for higher class pupils and ways to come up," *Inf. Technol. Teach. Aids*, vol. 66, no. 4, pp. 54–64, 2018.

[37] Y. Mumcu and S. Yıldız, "The investigation of algorithmic thinking skills of 5 th and 6 th graders according to different variables *," *MATDER J. Math. Educ.*, vol. 3, no. 1, pp. 41–48, 2018.

[38] M. Guerreo and J. García, "Algorithmic Thinking Development With Generative Learning Objects Support," *Pixel-Bit- Rev. Medios Y Educ.*, vol. 49, pp. 163–175, 2016, doi: 10.12795/pixelbit.2016.i49.011.

[39] M. Csernoch, P. Biró, J. Máth, and K. Abari, "Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments," *INFORMATICS Educ.*, vol. 14, no. 2, pp. 175–197, Oct. 2015, doi: 10.15388/infedu.2015.11.

[40] D. Knuth, *Algoritmos fundamentales. Volumen 1*. España: Reverte, s.a., 1980.

[41] J. Analoca, "Pensamiento Algorítmico en la Matemática de la Enseñanza Básica Algorithmic Thinking Mathematics in Basic Education," *Investig. y Tecnol.*, vol. 4, no. 1, pp. 1–12, 2016.

[42] G. Futschek, *Algorithmic Thinking: The Key for Understanding Computer Science*, vol. 4226, no. November. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[43] M. Christodoulou, E. Szczygieł, Ł. Kłapa, and W. Kolarz, *Algorithmic and Programming*, Krosno. Rzeszowska, Polonia, 2018.

[44] S. Passi and S. Jackson, "Data vision: Learning to see through algorithmic abstraction," in *CSCW'17 proceedings of the 2017 ACM conference on computer supported cooperative work and social computing. Portland, Oregon*, 2017, pp. 2436–2447.

[45] G. Brassard and P. Bratley, *Fundamentos de Algoritmia*. Madrid, España: Prentice Hall, 1998.

[46] Z. Katai, "The challenge of promoting algorithmic thinking of both sciences- and humanities-oriented learners," *J. Comput. Assist. Learn.*, vol. 31, no. 4, pp. 287–299, Aug. 2015, doi: 10.1111/jcal.12070.

[47] S. L. Thomas, D. Nafus, and J. Sherman, "Algorithms as fetish: Faith and possibility

in algorithmic work," *Big Data Soc.*, vol. 5, no. 1, pp. 1–10, Jun. 2018, doi: 10.1177/2053951717751552.

[48] H. Ramadhan, "Programming by discovery," *J. Comput. Assist. Learn.*, vol. 16, pp. 83–93, 2000.

[49] P. Dourish, "Algorithms and their others: Algorithmic culture in context," *Big Data Soc.*, vol. 3, no. 2, p. 205395171666512, Dec. 2016, doi: 10.1177/2053951716665128.

[50] RAE, "Algorítmo," *Real Academia Española*. 2019.

[51] D. X. Ramos Rivadeneira and J. A. Jiménez Toledo, "Incorporation of Algorithmic Thinking in Accounting Training," *Av. Investig. en Ing.*, vol. 16, no. 2, pp. 1–14, 2019, doi: 10.18041/1794-4953/avances.2.5445.

[52] R. Vinayakumar, K. Soman, and P. Menon, "Alg-Design: Facilitates to Learn Algorithmic Thinking for Beginners," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Jul. 2018, pp. 1–6, doi: 10.1109/ICCCNT.2018.8493952.

[53] J. Álvarez, "Pensamiento computacional y multimedia: un cambio en el paradigma educativo," 2013.

[54] A. I. Rincón Rueda and W. D. Ávila Díaz, "Una aproximación desde la lógica de la educación al pensamiento computacional," *Sophía*, vol. 2, no. 21, p. 161, Oct. 2016, doi: 10.17163/soph.n21.2016.07.

[55] P. Thagard, *La mente: introducción a las ciencias cognitivas*. Buenos Aires, Argentina: Katz Editores, 2008.

[56] J. Zapotecatl, *Introducción al pensamiento computacional: conceptos básicos para todos*. México D.F.: AmexComp, 2018.

[57] Google, "Exploring Computational Thinking," *Google for education*, 2020. https://edu.google.com/resources/programs/exploring-computational-thinking/# (accessed Jul. 25, 2020).

[58] M. Altaher and A. Ferchichi, "AlgoThink: An Algorithmic Computational Thinking Approach," in *2018 JCCO Joint International Conference on ICT in Education and Training, International Conference on Computing in Arabic, and International Conference on Geocomputing (JCCO: TICET-ICCA-GECO)*, Nov. 2018, pp. 1–7, doi: 10.1109/ICCA-TICET.2018.8726205.

[59] J. Wing, "Computational Thinking," *Comun. ACM*, vol. 49, no. 3, p. 35, 2006, [Online]. Available: https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf.

[60] F. Heintz, L. Mannila, and T. Farnqvist, "A review of models for introducing computational thinking, computer science and computing in K-12 education," in *2016 IEEE Frontiers in Education Conference (FIE)*, Oct. 2016, pp. 1–9, doi: 10.1109/FIE.2016.7757410.

[61] Y. Oomori *et al.*, "Algorithmic Expressions for Assessing Algorithmic Thinking Ability

of Elementary School Children," in *2019 IEEE Frontiers in Education Conference (FIE)*, Oct. 2019, pp. 1–8, doi: 10.1109/FIE43999.2019.9028486.

[62] C. Hu, "Computational thinking: What it might mean and what we might do about it.," in *Proceedings ofthe 16th Annual Joint Conference on Inno- vation and Technology in Computer Science Education*, 2011, pp. 223–227.

[63] S. Gökoğlu, "Algorithm Perception in Programming Education: A Metaphor Analysis," *Cumhur. Int. J. Educ.*, vol. 6, no. 1, pp. 1–14, 2017.

[64] G. Futschek, "Algorithmic thinking: The key for understanding computer science. In R. T. Mittermeir," in *ISSEP'06 Proceedings of the 2006 International Confer- ence on Informatics in Secondary Schools – evolution and perspectives: The bridge between using and understanding computers*, 2006, pp. 159–168.

[65] O. Vorobey, "Teaching Fundamentals of Algorithmization in Grade 5," *Komput. v shkoli ta simi*, vol. 2, pp. 7–10, 2014.

[66] C. Palma and R. Sarmiento, "Estado del arte sobre experiencias de enseñanza de programación a niños y jóvenes para el mejoramiento de las competencias matemáticas en primaria," *Rev. Mex. Investig. Educ.*, vol. 20, no. 65, pp. 607–641, 2015, [Online]. Available: http://www.redalyc.org/pdf/140/14035408013.pdf.

[67] I. Matyushchenko, E. Zvereva, and T. Lavina, "Development of Algorithmic Thinking by Means of Lego Mindstorms Ev3 on Robotics," in *2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, May 2020, pp. 444–447, doi: 10.1109/USBEREIT48449.2020.9117764.

[68] P. Compañ, R. Satorre, F. Llorens, and R. Molina, "Enseñando a programar: un camino directo para desarrollar el pensamiento computacional," *Rev. Educ. a Distancia*, vol. 46, no. 46, Sep. 2015, doi: 10.6018/red/46/11.

[69] S. Abramovich, "Mathematical problem posing as a link between algorithmic thinking and conceptual cnowledge," *Teach. Math.*, vol. XVIII, no. 2, pp. 45–60, 2015, [Online]. Available: http://elib.mi.sanu.ac.rs/files/journals/tm/35/tmn35p45-60.pdf.

[70] S. Grover, "Computer science is not just for big kids," *Learn. Lead. with Technol.*, vol. 37, no. 3, pp. 27–29, 2009.

[71] E. Lockwood, A. Asay, A. F. DeJarnette, and M. Thomas, "Algorithmic thinking: An initial characterization of computational thinking in mathematics," in *38th Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*, 2016, pp. 1588–1595.

[72] M. Guerrero Posadas and J. García Orozco, "Desarrollo Del Pensamiento Algorítmico Con El Apoyo De Objetos De Aprendizaje Generativos Algorithmic Thinking Development With Generative Learning Objects Support," *Píxel-Bit, Rev. Medios y Educ.*, no. 49, pp. 163–175, 2016, doi: 10.12795/pixelbit.2016.i49.11.

[73] E. Lamagna, "Algorithmic thinking unplugged," *J. Comput. Sci. Coll.*, vol. 30, no. 6,

pp. 45–52, 2015.

[74] R. Ruíz, *Una Introducción a la programación estructurada en C*, Primera ed. Santa Fe, Argentina: El Cid Editor, 2013.

[75] P. Halperin, “Development of research on the formation of mental actions,” *Psychol. Sci. USSR*, vol. 1, pp. 441–469, 1959.

[76] N. Talyzina, *Educational psychology*. Moscow: Center “Academy,” 2006.

[77] O. Romero, M. Rosero, and J. Jiménez, *La metacognición y la teoría de la actividad en la enseñanza de la programación*, Editorial. Colombia: Institución Universitaria CESMAG, 2017.

[78] L. Landa, *Algorithms and software training. Some questions of the theory and methodology of programming*. Moscow, Russi: Prosveshchenie, 1965.

[79] S. I. Malik, M. Shakir, A. Eldow, and M. W. Ashfaque, “Promoting Algorithmic Thinking in an Introductory Programming Course,” *Int. J. Emerg. Technol. Learn.*, vol. 14, no. 01, p. 84, Jan. 2019, doi: 10.3991/ijet.v14i01.9061.

[80] O. Arkhipov, “State and prospects of basic computer training in engineering education,” *Open Educ.*, vol. 20, no. 6, pp. 27–33, Jan. 2016, doi: 10.21686/1818-4243-2016-6-27-33.

[81] S. Altukhova and I. Smirnova, “Development of algorithmic thinking of university students in the process of professional and teacher training,” *Sci. notes Orel State Univ.*, vol. 2, no. 71, pp. 200–203, 2016.

[82] J. Hromkovič, T. Kohn, D. Komm, and G. Serafini, “Examples of Algorithmic Thinking in Programming Education,” *Olympiads in Informatics*, vol. 10, no. 1, pp. 111–124, Jul. 2016, doi: 10.15388/ioi.2016.08.

[83] P. Pintrich, R. Marx, and R. Boyle, “Beyond cold conceptual change: The role of motivational beliefs and classroom contextual factors in the process of concep- tual change.,” *Rev. Educ. Res.*, vol. 63, pp. 167– 199, 1993.

[84] T. P. Pushkaryeva, T. A. Stepanova, and V. V. Kalitina, “Didadtic tools for the students’ algorithmic thinking development,” *Educ. Sci. J.*, vol. 19, no. 9, pp. 126–143, Jan. 2017, doi: 10.17853/1994-5639-2017-9-126-143.

[85] E. Milková and A. Hùlková, “Algorithmic and logical thinking development: Base of programming skills,” *WSEAS Trans. Comput.*, vol. 12, no. 2, pp. 41–51, 2013.

[86] M. Resnick *et al.*, “Scratch: Programming for all,” *Commun. ACM*, vol. 52, no. 11, p. 60, Nov. 2009, doi: 10.1145/1592761.1592779.

[87] M. Zhaldak, “Informatics is a fundamental scientific discipline,” *Comput. Sch. Fam.*, vol. 2, pp. 39–43, 2010.

[88] M. Kurilov, “Once more about the axioms of programming and about teaching him,” *Artif. Intell.*, vol. 3, pp. 4–12, 2015.

[89] I. Minty, “Forms of training organization for the formation of competencies in programming,” *Pedagog. High. Middle Sch.*, vol. 41, pp. 91–96, 2014.

[90] O. Castelblanco, L. Donado, E. Gerlein, and E. Gonzalez, “KALA: Robotic Platform for Teaching Algorithmic Thinking to Children,” in *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, Aug. 2019, pp. 260–265, doi: 10.1109/DEVLRN.2019.8850694.

[91] J. Mezak and P. P. Papak, “Learning scenarios and encouraging algorithmic thinking,” in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2018, pp. 0760–0765, doi: 10.23919/MIPRO.2018.8400141.

[92] A. Solomon, “The Role of Spatial Representations in CS Teaching and CS Learning,” in *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Oct. 2019, pp. 237–238, doi: 10.1109/VLHCC.2019.8818785.

[93] K. M. Y. Law, V. C. S. Lee, and Y. T. Yu, “Learning motivation in e-learning facilitated computer programming courses,” *Comput. Educ.*, vol. 55, no. 1, pp. 218–228, 2010, doi: http://dx.doi.org/10.1016/j.compedu.2010.01.007.

[94] M. Juganaru, *Inroducción a la programación*, Patría S.A. México D.F., 2014.

[95] N. Couthinjo, *Introducción a la programación con Python, algorítmos y lógica de programación para principiantes*, Novatec, L. Sao Paulo, Brasil, 2016.

[96] O. Trejos Buriticá, Ivan, “Consideraciones sobre la evolución del pensamiento a partir de los paradigmas de programación de computadores,” *Tecnura*, vol. 16, no. 32, pp. 68–83, 2012.

[97] J. Villalobos and R. Casallas, *Fundmanetos de programación, aprendizaje activo basado en casos*, Universida. Bogotá D. C., 2019.

[98] M. Cedano, A. Cedano, J. Rubio, and A. Vega, *Fundmanetos de computación para ingenieros*, Universida. Mexico D.F., 2014.

[99] C. Li, Z., Plaue and E. Kraemer, “A Spirit of Camaraderie: The Impact of Pair Programming on Retention. Department of Computer Science- The University of Georgia Athens, GA USA.,” GA USA, 2012.

[100] O. Revelo Sánchez, C. Collazos Ordóñez, and J. Jiménez Toledo, “El trabajo colaborativo como estrategia didáctica para la enseñanza / aprendizaje de la programación : una revisión sistemática de literatura,” *TecnoLógicas*, vol. 21, no. 41, pp. 115–134, 2018, doi: https://doi.org/10.22430/issn.2256-5337.

[101] D. Preston, “Using collaborative learning research to enhance pair programming pedagogy,” *ACM SIGITE Newsl.*, vol. 3, no. 1, pp. 16–21, Jan. 2006, doi: 10.1145/1113378.1113381.

[102] N. Salleh, E. Mendes, and J. Grundy, “Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review,” *IEEE Trans.*

Softw. Eng., vol. 37, no. 4, pp. 509–525, Jul. 2011, doi: 10.1109/TSE.2010.59.

[103] G.-J. Hwang, Z.-Y. Liang, and H.-Y. Wang, “An Online Peer Assessment-Based Programming Approach to Improving Students’ Programming Knowledge and Skills,” in *2016 International Conference on Educational Innovation through Technology (EITT)*, Sep. 2016, pp. 81–85, doi: 10.1109/EITT.2016.23.

[104] J. P. Ucan, O. S. Gomez, and R. A. Aguilar, “Assessment of software defect detection efficiency and cost through an intelligent collaborative virtual environment,” *IEEE Lat. Am. Trans.*, vol. 14, no. 7, pp. 3364–3369, Jul. 2016, doi: 10.1109/TLA.2016.7587643.

[105] C. Collazos, L. Guerrero, and A. Vergara, “Aprendizaje Colaborativo: un cambio en el rol del profesor,” *Congr. Educ. Super. en Comput. Jornadas Chil. la Comput.*, p. p.1-10, 2001, [Online]. Available: https://users.dcc.uchile.cl/~luguerre/papers/CESC-01.pdf.

[106] E. Lovos, A. Gonzalez, I. Mouján, R. Bertone, and C. Madoz, “Estrategias de Enseñanza Colaborativa para un Curso de Programación de Primer Año de la Licenciatura en Sistemas,” 2012. [Online]. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/23850/Documento_completo.pdf?sequence=1.

[107] A. Nylén, N. Thota, A. Eckerdal, P. Kinnunen, M. Butler, and M. Morgan, “Multidimensional analysis of creative coding MOOC forums,” in *Proceedings of the 15th Koli Calling Conference on Computing Education Research - Koli Calling ’15*, 2015, pp. 137–141, doi: 10.1145/2828959.2828971.

[108] N. Moroni and P. Señas, “La Visualización de Algoritmos como Recurso para la Enseñanza de la Programación,” in *Workshop de Investigadores en Ciencias de la Computación. WICC 2002*, 2002, pp. 213–217.

[109] S. Sánchez *et al.*, “Applying Mixed Reality Techniques for the Visualization of Programs and Algorithms in a Programming Learning Environment,” in *eLmL 2018 : The Tenth International Conference on Mobile, Hybrid, and On-line Learning Applying*, 2018, no. c, pp. 84–89, [Online]. Available: https://www.thinkmind.org/index.php?view=article&articleid=elml_2018_8_30_50079.

[110] E. Costelloe, *Teaching Programming. The State of the Art*, Institute. Dublin, 2004.

[111] B. San Miguel, S. Aguirre, J. del Alamo, and M. Cortés, “A proposal for enhancing the motivation in students of computer programming,” in *CERI2012 Proceedings*, 2012, pp. 1157–1164, [Online]. Available: https://library.iated.org/view/SANMIGUEL2012APR.

[112] S. Naz, S. Hamad Shirazi, T. Iqbal, D. Irfan, M. Junaid, and Y. Naseer, “Learning Programming through Multimedia and Dry-run,” *Res. J. Appl. Sci. Eng. Technol.*, vol. 7, no. 21, pp. 4455–4463, Jun. 2014, doi: 10.19026/rjaset.7.822.

[113] S. Alhazbi, “Using e-journaling to improve self-regulated learning in introductory

computer programming course," in *2014 IEEE Global Engineering Education Conference (EDUCON)*, Apr. 2014, pp. 352–356, doi: 10.1109/EDUCON.2014.6826116.

[114] M. Guerrero, D. S. Guamán, and J. C. Caiza, "Revisión de Herramientas de Apoyo en el Proceso de Enseñanza- Aprendizaje de Programación," *Rev. Politécnica Nac.*, vol. 35, no. 1, pp. 84–90, 2015, [Online]. Available: https://revistapolitecnica.epn.edu.ec/ojs2/index.php/revista_politecnica2/article/view/430.

[115] I. Blanchette and K. Dunbar, "How analogies are generated: The roles of structural and superficialsimilarity.," *Mem. Cogn.*, vol. 28, pp. 108–124, 2000.

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

[117] R. T. Putnam, D. Sleeman, J. A. Baxter, and L. K. Kuspa, "A summary of misconceptions of high school Basic programmers," *J. Educ. Comput. Res.*, vol. 2, no. 4, pp. 459–472, 1986.

[118] D. Perez-Marin, R. Hijon-Neira, and M. Martin-Lope, "A Methodology Proposal Based on Metaphors to Teach Programming to Children," *IEEE Rev. Iberoam. Tecnol. del Aprendiz.*, vol. 13, no. 1, pp. 46–53, Feb. 2018, doi: 10.1109/RITA.2018.2809944.

[119] R. Muñoz *et al.*, "Uso de Scratch y Lego Mindstorms como Apoyo a la Docencia en Fundamentos de Programación," *XXI Jornadas la Enseñanza Univ. Informática*, pp. 248–254, 2015, [Online]. Available: http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2015/mu_usod.pdf.

[120] B. Restrepo Gomez, "Aprendizaje basado en problemas (ABP):una innovación didáctica para la enseñanza universitaria," *Educadores*, vol. 8, pp. 9–19, 2005.

[121] J. Jimenez, *CESMAG*, vol. VI, no. 2. Institucion Universitaria CESMAG, 2015.

[122] A. Robins, J. Rountree, and N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Comput. Sci. Educ.*, vol. 13, no. 2, pp. 137–172, Jun. 2003, doi: 10.1076/csed.13.2.137.14200.

[123] A. Pears *et al.*, "A survey of literature on the teaching of introductory programming," *SIGCSE Bull*, vol. 39, no. 4, pp. 204–223, 2007, doi: 10.1145 / 3293881.3295779.

[124] S. Wiedenbeck, V. Ramalingam, S. Sarasamma, and C. Corritore, "A comparison of the comprehension of object-oriented and procedural programs by novice programmers," *Interact. Comput.*, vol. 11, no. 3, pp. 255–282, Jan. 1999, doi: 10.1016/S0953-5438(98)00029-0.

[125] C. Patek and A. Chattopadhyay, "Can Undergraduate Computing Research Be Student-Driven? (Abstract Only)," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17*, 2017, pp. 715–715, doi: 10.1145/3017680.3022445.

[126] T. Koulouri, S. Lauria, and R. D. Macredie, “Teaching Introductory Programming,” *ACM Trans. Comput. Educ.*, vol. 14, no. 4, pp. 1–28, Dec. 2014, doi: 10.1145/2662412.

[127] L. Mannila and M. de Raadt, “An objective comparison of languages for teaching introductory programming,” in *Proceedings of the 6th Baltic Sea conference on Computing education research Koli Calling 2006 - Baltic Sea ’06*, 2006, p. 32, doi: 10.1145/1315803.1315811.

[128] S. Davies, J. A. Polack-Wahl, and K. Anewalt, “A snapshot of current practices in teaching the introductory programming sequence,” in *Proceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE ’11*, 2011, p. 625, doi: 10.1145/1953163.1953339.

[129] M. Rizvi, T. Humphries, D. Major, M. Jones, and H. Lauzun, “A CS0 course using Scratch,” *Comput. Sci. Coll.*, vol. 26, no. 3, pp. 19–27, 2011.

[130] C. Herbert, “An introduction to programming with Alice,” Boston, Massachusetts, 2007.

[131] T. Boyle, “Constructivism: A Suitable Pedagogy for Information and Computing Sciences?. Conference of the LTSN-ICS,” 2000, [Online]. Available: http://www.ics.ltsn.ac.uk/pub/conf2000/Papers/tboyle.htm.

[132] T. Crews and J. Butterfield, “Usingtechnology to bring abstract concepts into focus: A programming case study,” *Comput. High. Educ.*, vol. 13, no. 2, pp. 25–50, 2002, [Online]. Available: https://eric.ed.gov/?id=EJ647358.

[133] M. Carlisle, T. Wilson, J. Humphries, and S. Hadfield, “Raptor: A visual programming environment for teaching algorithmic problem solving,” *SIGCSE Bull*, vol. 37, no. 1, pp. 176–180, 2005, [Online]. Available: https://martincarlisle.com/publications/ccsc_named.pdf.

[134] A. Del Prado and N. Lamas, “Alternativas para la enseñanza de pseudocódigo y diagrama de flujo,” *Riect*, vol. 5, no. 3, pp. 102–113, 2014, [Online]. Available: http://www.exactas.unca.edu.ar/riecyt/VOL 5 NUM 3/F SI 3 14 Trabajo Completo Fundamentos.pdf.

[135] UEF, “Jeliot 3,” *Jeliot 3*, 2020. https://cs.joensuu.fi/jeliot/description.php.

[136] K. A. Reek, “The TRY System or How to Avoid Testing Students Programs,” in *Proceedings of SIGCSE*, 1989, pp. 112–116, doi: https://doi.org/10.1145/65294.71198.

[137] E. L. Jones, “Grading student programs a software testing approach,” *J. Comput. Sci. Coll.*, vol. 16, no. 2, 2001, [Online]. Available: https://dl.acm.org/citation.cfm?id=369354.

[138] L. Malmi, V. Karavirta, A. Korhonen, and J. Nikander, “Experiences on automatically assessed algorithm simulation exercises with different resubmission policies,” *J. Educ. Resour. Comput. (JERIC).*, vol. 5, no. 3, 2005, doi: https://doi.org/10.1145/1163405.1163412.

[139] C. Higgins, G. Gray, P. Symeonidis, and A. Tsintfsias, “Automated assessment and experiences of teaching programming,” *ACM J. Educ. Resour. Comput.*, vol. 5, no. 5, pp. 1–21, 2005, doi: https://doi.org/10.1145/1163405.1163410.

[140] T. Wang, X. Su, P. Ma, Y. Wang, and K. Wang, “Ability training oriented automated assessment in introductory programming course,” *J. Comput. Educ.*, vol. 56, no. 1, 2011, doi: https://doi.org/10.1016/j.compedu.2010.08.003.

[141] T. Barnes *et al.*, “The Role of Feedback,” in *Computer/Human Interaction 2007*, 2007, pp. 1–5.

[142] T. Barnes, A. Chaffin, E. Powell, and H. Lipford, “Game2Learn: Improving the motivation of CS1 students.,” in *Proceedings of the 3rd international conference on Game development in computer science education*, 2008, pp. 1–5.

[143] A. Chaffin, K. Doran, D. Hicks, and T. Barnes, “Experimental evaluation of teaching recursion in a video game,” in *Proceedings ACM SIGGRAPH Symposium on Video Games*, 2009, pp. 79–86.

[144] M. Eagle and T. Barnes, “Experimental evaluation of an educational game for improved learning in introductory computing,” *SIGCSE Bull*, vol. 41, no. 1, pp. 321–325, 2009, doi: doi.org/10.1145/1508865.1508980.

[145] F. W. B. Li and C. Watson, “Game-based concept visualization for learning programming,” in *Proceedings of the third international ACM workshop on Multimedia technologies for distance learning*, 2011, pp. 37–42, doi: https://doi.org/10.1145/2072598.2072607.

[146] M. Piteira and S. Haddad, “Innovate in Your Program Computer Class: An approach based on a serious game,” 2011.

[147] K. Maragos and M. Grigoriadou, “Exploiting TALENT as a Tool for Teaching and Learning,” *Int. J. Learn.*, vol. 18, no. 1, pp. 431–440, 2011, [Online]. Available: https://www.researchgate.net/profile/Maya_Satratzemi/publication/266078077_Educational_games_for_computer_programming/links/570f7a6c08ae170055bc4419/Educational-games-for-computer-programming.pdf.

[148] M. Lee and A. Ko, “Personifying programming tool feedback improves novice programmers’ learning,” in *Proceedings of the Seventh International Workshop on Computing Education Research, Providence: ACM*, 2011, pp. 109–116, doi: 10.1145/2016911.2016934.

[149] J. O’Kelly and P. Gibson, “RoboCode & problem-based learning: A non-prescriptive approach to teaching programming,” *ACM SIGCSE Bull.*, vol. 38, no. 3, pp. 217–221, 2006, doi: 10.1145/1140123.1140182.

[150] A. Phelps, K. Bierre, and D. Parks, “MUPPETS: multi-user programming pedagogy for enhancing traditional study,” in *Proceeding of the 4th conference on Information technology education*, 2004, pp. 100–105, doi: https://dx.doi.org/10.1109/FIE.2005.1612247.

[151] M. Muratet, P. Torguet, F. Viallet, and J. P. Jessel, “Experimental Feedback on

Prog&Play: A Serious Game for Programming Practice," *Comput. Graph. Forum*, vol. 30, no. 1, pp. 61–73, Mar. 2011, doi: 10.1111/j.1467-8659.2010.01829.x.

[152] I. Paliokas, C. Arapidis, and M. Mpimpitsos, "PlayLOGO 3D: A 3D Interactive Video Game for Early Programming Education: Let LOGO Be a Game," in *2011 Third International Conference on Games and Virtual Worlds for Serious Applications*, May 2011, pp. 24–31, doi: 10.1109/VS-GAMES.2011.10.

[153] M. J. Lee and A. J. Ko, "Personifying Programming Tool Feedback Improves Novice Pr ogrammers' Learning," in *Conference on International Computing Education Research (ICER)*, 2011, pp. 109–116.

[154] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," 2012, [Online]. Available: http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf.

[155] R. Zúñiga, J. Hurtado, and P. Paderewsky, "Discovering the mechanisms of abstraction in the performance of work teams in children to solve computational problems," *Sist. Telemática*, vol. 14, no. 36, pp. 69–87, 2016, doi: 10.18046/syt.v14i36.2216.

[156] J. Mönig, "Snap: Build your own blocks," *Snap*, 2015. https://snap.berkeley.edu/.

[157] C. C. Liu, Y. B. Cheng, and C. W. Huang, "The effect of simulation games on the learning of computational problem solving," *Comput. Educ.*, vol. 57, no. 3, pp. 1907–1918, 2011, doi: 10.1016/j.compedu.2011.04.002.

[158] J. García, P. Señas, and N. Moroni, "Cubik: Una Herramienta de Apoyo a la Enseñanza de la Programación," 1996.

[159] N. Moroni, "Entornos para el aprendizaje de la programación," Bahía Blanca (Argentina), 2015.

[160] G. M. and F. Gortázar, "EclipseGavab, un entorno de desarrollo para la docencia online de la programación," 2009.

[161] J. C. Rodriguez del Pino, E. Royo Rubio, and F. Hernandez Figueroa, "VPL: Laboratorio virtual de programación para Moodle," in *XVI Jornadas de Enseñanza Universitaria de Informática, Jenui .*, 2010, pp. 429–435.

[162] J. A. Jiménez Builes, M. Pavony Meneses, and A. F. Alvarez Serna, "Entorno de integración de PBL y CSCL para la enseñanza de algoritmos y programación en ingeniería," *Av. en Sist. e Informática*, vol. 5, no. 3, pp. 189-194., 2008, [Online]. Available: https://revistas.unal.edu.co/index.php/avances/article/view/10112/10637.

[163] M. Á. Redondo and M. Ortega, "Colecce 2.0," *¿Qué es COLLECE 2.0?*, 2018. http://blog.uclm.es/grupochico/proyecto-iapro/collece-2-0/.

[164] "Real Academia Española: Diccionario de la lengua española," *23.ª ed.*, 2021. https://dle.rae.es (accessed Sep. 22, 2021).

[165] J. Tristán Fernández *et al.*, “Entorno y desarrollo,” *Rev. Pediatría Atención Primaria*, vol. 9, no. 36, pp. 623–634, 2007.

[166] M. Prada, “Entre las máquinas y los entornos: la idea de tecnología para la enseñanza de la filosofía en la posmodernidad,” *Pedagog. y Saberes*, no. 31, 2009, doi: 10.17227/01212494.31pys44.50.

[167] Euroinnova, “Entorno educativo,” *Online*, 2021. https://www.euroinnova.co/blog/entorno-educativo#iquestqueacute-es-el-entorno-educativo (accessed Sep. 22, 2021).

[168] Gabetta, “El entorno como elemento educativo,” *NBER Work. Pap. Ser.*, vol. 58, no. 58, pp. 99–104, 1989, [Online]. Available: https://www.unhcr.org/publications/manuals/4d9352319/unhcr-protection-training-manual-european-border-entry-officials-2-legal.html?query=excom 1989.

[169] E. Pirttiniemi and A. Rouvari, “Dimensión didáctica del entrono de aprendizaje.” http://www.cibernarium.tamk.fi/havainnollistaminen_es/didactic_.

[170] A. Chiechier, *Entornos virtuales y aprendizaje. Nuevas perspectivas de estudio e investigaciones*, 1st ed., no. 1. Mendoza, Argentina: Editorial Virtual Argentina, 2013.

[171] Pressboks, “¿Qué es un entorno de aprendizaje?,” *El Glosario de la Reforma Educativa, 29 de agosto de 2014*, 2014. https://cead.pressbooks.com/chapter/a-2-que-es-un-entorno-de-aprendizaje/.

[172] F. Pavón and J. Casanova, “Francisco Pavón Rabasco Juan Casanova Correa Como profesores de Universidad , nos estamos preparando para adaptar la impartición de nuestras asignaturas a las orientaciones que nos dan de cara al,” *RIED. Rev. Iberoam. Educ. a Distancia*, vol. 10, no. 2, pp. 148–163, 2007.

[173] A. Gonzalez, “Entornos de programación,” *Academia*, 2017. https://www.academia.edu/34901007/Practica_Entornos_de_programación.

[174] S. Glynn, B. Brillan, M. Semrud Clikman, and K. Muth, “Analogical reasoning and problem solving in Science,” in *Textbooks. HandbookofCreativity*, 1989, pp. 383–398, [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4757-5356-1_21.

[175] C. Arca, “Comparaciones, metáforas, analogías,” La Plata (Argentina), 2015. [Online]. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/68742/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y.

[176] L. Haaparanta, “The Analogy Theory of Thinking. In:,” *Dialectica*, vol. 46, no. 2, pp. 169–183, 1992, [Online]. Available: https://www.jstor.org/stable/42969049%0A.

[177] C. Perelman, *Analogie et Metaphore en Science, Poesie et Philosophie.* Bruxelles: Presses Universitaires de Bruxelles., 1970.

[178] J. M. Oliva, “Actividades para la enseñanza/aprendizaje de la química a través de

analogías," *Rev. Eureka sobre enseñanza y Divulg. las ciencias*, vol. 3, no. 1, pp. 104–114, 2006, doi: 10.25267/Rev_Eureka_ensen_divulg_cienc.2006.v3.i1.08.

[179] J. K. Gilbert, *Multiple Representations in Chemical Education*, vol. 4. Dordrecht: Springer Netherlands, 2009.

[180] A. G. Harrison and D. F. Treagust, "A typology of school science models," *Int. J. Sci. Educ.*, vol. 22, no. 9, pp. 1011–1026, Sep. 2000, doi: 10.1080/095006900416884.

[181] L. Galagovsky and A. Adúriz-Bravo, "Models and analogies in the teaching of natural sciences. The concept of analogical didactic model," *Enseñanza las Ciencias*, vol. 19, no. 2, pp. 231–242, 2001, [Online]. Available: https://www.raco.cat/index.php/Ensenanza/article/view/21735.

[182] J. Fernández González, B. M. González González, and T. Moreno Jiménez, "Considerations about research in analogies," *Estud. Front.*, vol. 5, no. 9, pp. 79–105, 2004, [Online]. Available: http://www.scielo.org.mx/pdf/estfro/v5n9/v5n9a4.pdf.

[183] N. Owen, *La magia de la metáfora : 77 relatos breves para educadores, formadores y pensadores*, no. Book, Whole. 2003.

[184] RAE, "Metáfora," *Real académia Española, RAE*, 2018. .

[185] D. Vázquez, "Metáfora y Analogía,Su distinción y uso en la ciencia y la filosofía," *Tópicos*, vol. 38, pp. 85–117, 2010.

[186] J. P. Sanford, A. Tietz, S. Farooq, S. Guyer, and R. B. Shapiro, "Metaphors we teach by," in *In Proceedings of the 45th ACM technical symposium on Computer science education*, 2014, pp. 585–590, [Online]. Available: https://dl.acm.org/citation.cfm?id=2538862.

[187] H. Rodriguez, "Metáforas y analogías: diferencias y ejemplos [2021]," *Crehana*, 2021. https://www.crehana.com/co/blog/estilo-vida/metaforas-analogias/ (accessed Oct. 01, 2021).

[188] Ó. Castillero, "Diferencias entre metáfora, analogía y alegoría," *Psicología y mente*, 2021. https://psicologiaymente.com/cultura/diferencias-entre-metafora-analogia-alegoria.

[189] F. J. Ruiz and C. Luciano, "Relating relationships as a functional analytical model of analogy and metaphor," *Acta Comport.*, vol. 20, pp. 3–29, 2012, [Online]. Available: http://www.revistas.unam.mx/index.php/acom/article/view/35536/32356.

[190] A. Adúriz-Bravo, J. Garófalo, M. Greco, and L. Galagovsky, "Modelo didáctico analógico: Marco teórico y ejemplos," in *Enseñanza de las Ciencias*, 2005, pp. 1–6, [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Modelo+did?ctico+anal?gico.+marco+te?rico+y+ejemplos#0.

[191] J. Marmalejos, E. Paulino, and R. Gómez, "Propuesta de estrategias que fomentan

el aprendizaje y la solución de problemas en las ciencias básicas fortaleciendo la interpretación y aplicación del despeje, la sustitución numérica en ecuaciones y formulas, para los estudiantes del ciclo básico de la," in *Ciencia, Congreso Iberoamericano De*, 2014, pp. 1–26.

[192] R. Zamorano, H. Gibbs, L. Moro, and J. Viau, "Evaluación de un modelo didáctico analógico para el aprendizaje de energía interna y temperatura," *Eureka sobre Enseñanza y Divulg. las Ciencias*, vol. 3, no. 3, pp. 392–408, 2006.

[193] M. E. I. Malachías and D. Borges dos Santos, "Aprendizagem Significativa Crítica pela proposição explicativa de analogias através do Modelo Didático Analógico (MDA)," *Rev. Electrónica Investig. en Educ. en Ciencias, vol.*, vol. 8, no. 2, pp. 21–33, 2013, [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=5119863.

[194] F. González, G. González, M. Benigno, and M. Jiménez, "Towards an evolution of the conception of analogy," *Sci. Teach.*, vol. 23, no. 1, pp. 33–45, 2005, [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=1103653.

[195] L. R. Galagovsky and A. Adúriz-Bravo, "Modelos y analogías en la enseñanza de las ciencias naturales. El concepto de modelo didáctico analógico," *Enseñanza las Ciencias. Rev. Investig. y Exp. didácticas*, vol. 19, no. 2, p. 231, 2001, doi: 10.5565/rev/ensciencias.4000.

[196] F. G. Arrese, J. L. Olivares, M. Villarreal, N. G. Vincet, and V. Alfageme, "Modelo didáctico analógico como mediador de enseñanza y aprendizaje universitario del Sistema Cardiovascular," *Rev. Eureka sobre Enseñanza y Divulg. las Ciencias*, vol. 17, no. 3, pp. 1–15, 2020, doi: 10.25267/rev_eureka_ensen_divulg_cienc.2020.v17.i3.3601.

[197] C. D. Hundhausen, S. A. Douglas, and J. T. Stasko, "A metastudy of algorithm visualization effectiveness," *J. Vis. Lang. Comput.*, vol. 13, pp. 259–290, 2002, doi: 10.1006/S1045-926X(02)00028-9.

[198] C. Lacave, M. A. Garcia, A. I. Molina, S. Sanchez, M. A. Redondo, and M. Ortega, "COLLECE-2.0: A real-time collaborative programming system on Eclipse," in *2019 International Symposium on Computers in Education (SIIE)*, Nov. 2019, pp. 1–6, doi: 10.1109/SIIE48397.2019.8970132.

[199] R. Sukamto and R. Megasari, "Analogy mapping for different learning style of learners in programming," in *2017 3rd International Conference on Science in Information Technology (ICSITech)*, Oct. 2017, pp. 626–631, doi: 10.1109/ICSITech.2017.8257189.

[200] K. Strunz and H. Louie, "Cache Energy Control for Storage: Power System Integration and Education Based on Analogies Derived From Computer Engineering," *IEEE Trans. Power Syst.*, vol. 24, no. 1, pp. 12–19, Feb. 2009, doi: 10.1109/TPWRS.2008.2005713.

[201] A. Plappally, "The effect of joint role of creative analogy and concept-in-context map on the learning interest and performance of first year mechanical engineering

undergraduates," in *2016 IEEE 8th International Conference on Engineering Education (ICEED)*, Dec. 2016, pp. 126–130, doi: 10.1109/ICEED.2016.7856057.

[202] A. Stockdill *et al.*, "Correspondence-based analogies for choosing problem representations," in *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Aug. 2020, pp. 1–5, doi: 10.1109/VL/HCC50065.2020.9127258.

[203] L. Arias, "Situated learning and cognitive development," *Simon Bolivar University*, 2011. https://docplayer.es/27825700-El-aprendizaje-situado-y-el-desarrollo-cognitivo-comparacion-entre-las-teorias-aprendizaje-situado-y-desarrollo-cognitivo-de-brunner.html.

[204] E. Cruz, "Situated learning and meaningful learning," 2011, [Online]. Available: http://esther-upnesthercriteriospas.blogspot.com/2011/02/el-aprendizaje-situado-y-aprendizaje.html.

[205] J. R. Anderson, L. M. Reder, and H. A. Simon, "Situated Learning and Education Situated Learning and Education'," *Source Educ. Res.*, vol. 25, no. 4, pp. 5–11, 1996, doi: 10.2307/1176775.

[206] M. Lobur, A. Romanyuk, and M. Romanyshyn, "Using NLTK for educational and scientific purposes," *IEEE Xplore*, vol. 1, p. 43, 2011, [Online]. Available: Using NLTK for educational and scientific purposes%0A.

[207] L. Xu, S. Sun, and Q. Wang, "Text similarity algorithm based on semantic vector space model," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, Jun. 2016, pp. 1–4, doi: 10.1109/ICIS.2016.7550928.

[208] C. Torres and L. Arco, "Textual representation in semantic vector space," *Rev. Cuba. Ciencias Informáticas*, vol. 10, no. 2, pp. 148–180, 2016, [Online]. Available: https://rcci.uci.cu/?journal=rcci&page=article&op=view&path%5B%5D=1236.

[209] J.-K. Kim, G. Tur, A. Celikyilmaz, B. Cao, and Y.-Y. Wang, "Intent detection using semantically enriched word embeddings," in *2016 IEEE Spoken Language Technology Workshop (SLT)*, Dec. 2016, pp. 414–419, doi: 10.1109/SLT.2016.7846297.

[210] N. Sano, "Synthetic Data by Principal Component Analysis," in *2020 International Conference on Data Mining Workshops (ICDMW)*, Nov. 2020, pp. 101–105, doi: 10.1109/ICDMW51313.2020.00023.

[211] S. Yadav and S. Shukla, "Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification," in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, Feb. 2016, no. Cv, pp. 78–83, doi: 10.1109/IACC.2016.25.

[212] N. Sedaghat, M. Fathy, M. H. Modarressi, and A. Shojaie, "Combining Supervised and Unsupervised Learning for Improved miRNA Target Prediction," *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, pp. 1–1, 2018, doi: 10.1109/TCBB.2017.2727042.

[213] D. Cedeno and M. Vargas, "Application of Machine Learning with Supervised

Classification Algorithms: In the Context of Health," in *2019 7th International Engineering, Sciences and Technology Conference (IESTEC)*, Oct. 2019, pp. 613–618, doi: 10.1109/IESTEC46403.2019.00115.

[214] R. Nijhawan, I. Srivastava, and P. Shukla, "Land cover classification using supervised and unsupervised learning techniques," in *2017 International Conference on Computational Intelligence in Data Science(ICCIDS)*, Jun. 2017, pp. 1–6, doi: 10.1109/ICCIDS.2017.8272630.

[215] B. Karacali, "Improved quasi-supervised learning by expectation-maximization," in *2013 21st Signal Processing and Communications Applications Conference (SIU)*, Apr. 2013, pp. 1–4, doi: 10.1109/SIU.2013.6531366.

[216] V. Lytvyn, V. Vysotska, O. Veres, I. Rishnyak, and H. Rishnyak, "Content linguistic analysis methods for textual documents classification," in *2016 XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT)*, Sep. 2016, pp. 190–192, doi: 10.1109/STC-CSIT.2016.7589903.

[217] G. Weir, K. Owoeye, A. Oberacker, and H. Alshahrani, "Cloud-based Textual Analysis as a Basis for Document Classification," in *2018 International Conference on High Performance Computing & Simulation (HPCS)*, Jul. 2018, pp. 672–676, doi: 10.1109/HPCS.2018.00110.

[218] D. Yun, W. Liu, C. Q. Wu, N. S. V Rao, and R. Kettimuthu, "Performance Prediction of Big Data Transfer Through Experimental Analysis and Machine Learning," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 181–189, [Online]. Available: https://ieeexplore.ieee.org/document/9142699.

[219] P. Gamallo and M. García, "Methods on Natural Language Processing for Information Retrieval," Universidade de Santiago de Compostela, Chile, 2012. [Online]. Available: https://gramatica.usc.es/~gamallo/artigos-web/Novativa2012.pdf.

[220] Y. Pérez-Guadarramas, A. Rodríguez-Blanco, A. Simón-Cuevas, W. Hojas-Mazo, and J. Á. Olivas, "Combining lexical - syntactic patterns and topic analysis for automatic keyphrase extraction from texts," *Proces. del Leng. Nat.*, vol. 59, pp. 39–46, 2017, [Online]. Available: http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/5491/3250.

[221] A. Rodriguez-Blanco, A. Simon-Cuevas, W. Hojas-Mazo, and J. Perea-Ortega, "Extraction of linked data from unstructured information applying NLP techniques and ontologies," *CEUR Workshop Proc.*, vol. 1797, no. February 2017, pp. 80–91, 2016, [Online]. Available: https://www.researchgate.net/publication/312730295_Extraccion_de_datos_enlazados_desde_informacion_no_estructurada_aplicando_tecnicas_de_PLN_y_ontologias.

[222] D. Perez-Marin, R. Hijon-Neira, and M. Martin-Lope, "A Methodology Proposal Based on Metaphors to Teach Programming to Children," *IEEE Rev. Iberoam.*

Tecnol. del Aprendiz., vol. 13, no. 1, pp. 46–53, Feb. 2018, doi: 10.1109/RITA.2018.2809944.

[223] D. Pérez-Marín, R. Hijón-Neira, A. Bacelo, and C. Pizarro, “Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children?,” *Comput. Human Behav.*, vol. 105, p. 105849, Apr. 2020, doi: 10.1016/j.chb.2018.12.027.

[224] C. Chibaya, “A Metaphor-Based Approach for Introducing Programming Concepts,” in *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, Nov. 2019, pp. 1–8, doi: 10.1109/IMITEC45504.2019.9015888.

[225] Weka, “Weka 3 - Data Mining with Open Source Machine Learning Software in Java,” *Weka home page*, 2020. https://www.cs.waikato.ac.nz/~ml/weka/ (accessed Oct. 15, 2021).

[226] R. Sanchez, “T-Student, usos y abusos,” *Rev. Mex. Cardiol.*, vol. 26, no. 1, pp. 59–61, 2015, [Online]. Available: https://www.medigraphic.com/cgi-bin/new/resumen.cgi?IDARTICULO=56921.

[227] RealEye, “Webcam Eye-Tracking basado en pantalla. en línea. | RealEye.io,” *Webcam en línea con productos de seguimiento ocular*, 2021. https://www.realeye.io/es/ (accessed Nov. 01, 2021).

[228] J. Nielsen, “Heuristic Evaluation: How-To: Article by Jakob Nielsen,” *How to Conduct a Heuristic Evaluation*, 1994. https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/ (accessed Oct. 24, 2021).

[229] D. Travis, “247 web usability guidelines,” *Userfocus*, 2016. https://www.userfocus.co.uk/resources/guidelines.html.

ANEXOS

Anexo 1

Revisión sistemática de literatura

Estructura de la revisión sistemática de literatura

Término principal	Resultado de la búsqueda	Archivos duplicados	Archivos Excluidos	Archivos Pertinentes	Base de datos
Teaching-learning of computer programming	48	11	19	18	Redalyc
	27	3	13	11	IEEE Xplorer
	8	0	1	7	Springer
	23	3	9	11	ISI (Web of Science)
Total	**106**	**17**	**42**	**46**	

Evaluación de calidad en procesos de búsqueda y selección

Año	Cantidad	Referencias
2012	9	[9] [15] [21] [27] [41] [73] [80] [86] [104]
2013	5	[29] [33] [35] [36] [123]
2014	9	[1] [38] [39] [40] [51] [98] [105] [114] [115]
2015	10	[12] [23] [34] [67] [75] [78] [92] [109] [111] [112]
2016	6	[14] [19] [48] [74] [89] [90]
2017	2	[4] [91]
2018	5	[47] [52] [53] [83] [107]
Total	**46**	

Estudios incluidos en revisión sistemática

Anexo 2

Formato de configuración inicial

Universidad del Cauca

Grupo de investigación IDIS

Formato de configuración inicial

Universidad			
Programa académico			
Nombre del curso		Semestre	
Profesor			

Analogías			
Concepto a enseñar			
Analogías para enseñanza concepto fundamental			
No	Verbo (Palabra clave)	Sustantivo	Contexto
1			
2			
3			
4			
5			
Observaciones:			
Analogías para enseñanza del ejemplo computacional fundamental			
No	Verbo (Palabra clave)	Sustantivo	Contexto
1			
2			
3			
4			
5			
Observaciones:			
Ejercicios complementarios computacionales			
No	Verbo (Palabra clave)	Sustantivo	Contexto
1			
2			
3			
4			
5			
Observaciones:			

Anexo 3

Formato test aptitud lógico matemático y pensamiento crítico

GRUPO DE INVESTIGACIÓN IDIS
GRUPO DE INVESTIGACIÓN TECNOFILIA

ASPIRANTES A INGENIERIA DE SISTEMAS
PRUEBA DE APTITUD LÓGICO-MATEMÁTICA

TECNOFILIA

Nombres: _____ **Identificación:** _____ **Fecha:** _____

Marque con una X la respuesta correcta

1. Juan se acostó a las 8 pm. Y puso el despertador a las 9 am. Cuanto tiempo duerme?
a) 13 horas b) 9 horas c) 12 horas
d) 1 hora

2. El costo de una boleta pasó de $ 800 a $ 1200 el porcentaje en que se incrementó fue:
a) 50% b) 150% c) 100%
d) 200%

3. Cuando x^0 - 1 se divide entre x - 1 el residuo es:
a) 0 b) 1 c) 3 d) 2

4. En un triangulo rectángulo su hipotenusa mide 5 cm y un cateto 3 cm, Su área será:
a) 6 b) 7.5 c) 8 d) 15

5. Hay 10 personas en un ascensor, cuatro mujeres y seis hombres. El peso promedio de las mujeres es de 60 Kilos y el de los hombres 80 kilos. ¿Cuál es el peso promedio de las 10 personas del ascensor?:
a) 70 Kg. b) 71 Kg. c) 75 Kg. d) 74 Kg.

6. En una caja de caramelos, la mitad son de chocolate, la cuarta parte son de arequipe, la sexta parte son de mora y el resto de coco. Si en total hay 18 caramelos de chocolate. ¿Cuántos son de coco?:
a) 3 b) 6 c) 5 d) 4

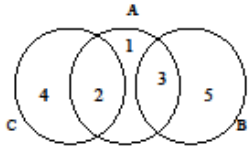
7. El salario de Jaime es cuatro veces mayor que el de Santiago. Si la suma de ambos sueldos es de $5.250.000. ¿Cuál será el sueldo de Jaime?:
a) $ 4.200.000 b) $ 5.000.000
c) $ 5.050.000 d) $ 4.050.000

8. Si de la igualdad 1/x - 1/y = 1/z se despeja x, resulta x igual a:

a) zy/(z+y) b) zy/(y-z) c) yz/(z-y)
d) (z+ y)/zy

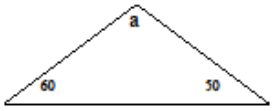
9. La expresión $(6a^3 - 3a)/3a^3$ es igual a:
a) $2 - a^{-2}$ b) $2a - 1/3a$ c) $a - 1/a$ d) $3a - a/a^2$

10. De acuerdo con el diagrama, es cierto que:



a) $A \cap B = \{3\}$ b) $B = \{2,5\}$ c) $A = \{1,4\}$
d) $C = \{4,3\}$

11. El valor del ángulo a en el triángulo, es.



a) 70° b) 80° c) 90° d) 60°

12. Las raíces de una ecuación de segundo grado son 3 y −5 la ecuación es:
a) $x^2 + 2x - 15 = 0$ b) $x^2 - 2x - 3 = 0$
c) $x^2 - 2x - 15 = 0$ d) $x^2 - 2x + 15 = 0$

13. La solución de: $\sqrt{X+2} - \sqrt{2X-3} = 0$ es:
a) 5 b) 2 c) 4 d) 3

14. Al introducir los tres últimos términos del polinomio $x^2 - a^2 + 2ab - b^2$ en un paréntesis se obtiene:

a) $x^2 + (a^2 - 2ab + b^2)$
b) $x^2 - (a^2 - 2ab + b^2)$
c) $x^2 + (-a^2 - 2ab + b^2)$
d) $x^2 - (-a^2 - 2ab + b^2)$

GRUPO DE INVESTIGACIÓN IDIS
GRUPO DE INVESTIGACIÓN TECNOFILIA

TECNOFILIA

ASPIRANTES A INGENIERIA DE SISTEMAS
PRUEBA DE COMPRENSION LECTORA

Nombres: _____ **Identificación:** _____ **Fecha:** _____

Marque con una X la respuesta correcta

EL ECLIPSE

Cuando Fray Bartolomé Arrazola se sintió perdido aceptó que ya nada podría salvarlo. La selva poderosa de Guatemala lo había apresado, implacable y definitiva. Ante su ignorancia topográfica se sentó con tranquilidad a esperar la muerte. Quiso morir allí, sin ninguna esperanza, aislado, con el pensamiento fijo en la España distante, particularmente en el convento de Los Abrojos, donde Carlos Quinto condescendiera una vez a bajar de su eminencia para decirle que confiaba en el celo religioso de su labor redentora.

Al despertar se encontró rodeado por un grupo de indígenas de rostro impasible que se disponía a sacrificarlo ante un altar, un altar que a Bartolomé le pareció como el lecho en que descansaría, al fin, de sus temores, de su destino, de sí mismo.

Tres años en el país le habían conferido un mediano dominio de las lenguas nativas. Intentó algo. Dijo algunas palabras que fueron comprendidas.

Entonces floreció en él una idea que tuvo por digna de su talento y de su cultura universal y de su arduo conocimiento de Aristóteles. Recordó que para ese día se esperaba un eclipse total de sol. Y dispuso, en lo más íntimo, valerse de aquel conocimiento para engañar a sus opresores y salvar la vida. -Si me matáis -les dijo- puedo hacer que el sol se oscurezca en su altura.

Los indígenas, lo miraron fijamente y Bartolomé sorprendió la incredulidad en sus ojos. Vio que se produjo un pequeño consejo, y esperó confiado, no sin cierto desdén.

Dos horas después el corazón de Fray Bartolomé Arrazola chorreaba su sangre vehemente sobre la piedra de los sacrificios (brillante bajo la opaca luz de un sol eclipsado), mientras uno de los indígenas recitaba sin ninguna inflexión de voz, sin prisa, una por una las infinitas fechas en que se producirían eclipses solares y lunares, que los astrónomos de la comunidad maya habían previsto y anotado en sus códices sin la valiosa ayuda de Aristóteles.

(Augusto Monterroso. Obras completas y otros cuentos, Bogotá, Norma, 1994-).

Anexo 4

Formato encuesta caracterización de analogías en CS1

Proyecto: Entornos de aprendizaje analógicos para el desarrollo de pensamiento algorítmico en la enseñanza de los fundamentos de programación

Encuesta dirigida a profesores de un primer curso de programación

Objetivo: Caracterizar los procesos analógicos formales y no formales desarrollados en clase para un primer curso de programación

Analogías: son comparaciones entre dos tópicos: uno nuevo (blanco) y otro conocido (tópico) y que se utilizan como herramientas del lenguaje para acercar a los estudiantes del conocimiento tradicional al conocimiento científico [194]. “trabajar con analogías implica, de alguna forma, una labor semejante al uso y construcción de modelos, por lo que implica de tarea la búsqueda de conexiones entre objetos, atributos y relaciones entre ellos. Implica, por tanto, una cierta sistematicidad de pensamiento, un argumentar razones a favor y en contra como hemos visto, y con ello también una forma diferente de ver el mundo, orientada desde criterios lógicos que van más allá del pensamiento implícito de sentido común” [178].

Datos de la Universidad	
Universidad	
País	
Tipo de Universidad	Pública:_____ Privada:_____
Datos Profesor	
Nombre	
Cargo	
Último Nivel de estudio	
Datos Curso	
Nombre	
Carrera	

1. ¿Ha utilizado usted analogías en su ejercicio docente, en cualquiera de los cursos a su cargo?
   Si____
   No____

2. ¿Ha utilizado usted analogías para la enseñanza de los conceptos básicos en un primer curso de programación?
   Si____
   No___

3. ¿Ha utilizado analogías para el concepto de Entrada/Salida?
   Si____
   No___

   Si su respuesta es “Si” por favor menciónalas:

No	Analogía (Nombre)	Concepto (Entradas o Salidas)
1		
2		
3		

Si su respuesta es “No” ¿puede por favor pensar en algunas analogías que en el contexto de sus estudiantes se puedan utilizar para dar a conocer estos conceptos?

No	Analogía (Nombre)	Concepto
1		**Entradas**
2		**Entradas**
3		**Salidas**
4		**Salidas**

4. ¿Ha utilizado analogías para el concepto de Condicional Simple?

   Si____
   No___

   Si su respuesta es “Si” por favor menciónalas:

No	Analogía (Nombre)
1	
2	
3	

   Si su respuesta es “No” ¿puede por favor pensar en algunas analogías que en el contexto de sus estudiantes se puedan utilizar para dar a conocer estos conceptos?

No	Analogía (Nombre)
1	
2	
3	

5. ¿Ha utilizado analogías para el concepto de Condicional Compuesto?

   Si____
   No___

   Si su respuesta es “Si” por favor menciónalas:

No	Analogía (Nombre)
1	
2	
3	

Si su respuesta es “No” ¿puede por favor pensar en algunas analogías que en el contexto de sus estudiantes se puedan utilizar para dar a conocer estos conceptos?

No	Analogía (Nombre)
1	
2	
3	

6. ¿Ha utilizado analogías para el concepto de Condicional Anidado?

Si____
No___

Si su respuesta es “Si” por favor menciónalas:

No	Analogía (Nombre)
1	
2	
3	

Si su respuesta es “No” ¿puede por favor pensar en algunas analogías que en el contexto de sus estudiantes se puedan utilizar para dar a conocer estos conceptos?

No	Analogía (Nombre)
1	
2	
3	

7. ¿Ha utilizado analogías para el concepto de Estructura Selectiva Múltiple?

Si____
No___

Si su respuesta es “Si” por favor menciónalas:

No	Analogía (Nombre)
1	
2	
3	

Si su respuesta es “No” ¿puede por favor pensar en algunas analogías que en el contexto de sus estudiantes se puedan utilizar para dar a conocer estos conceptos?

No	Analogía (Nombre)
1	
2	
3	

8. ¿Ha utilizado analogías para el concepto de Ciclo Para?

   Si____
   No___

   Si su respuesta es “Si” por favor menciónalas:

No	Analogía (Nombre)
1	
2	
3	

   Si su respuesta es “No” ¿puede por favor pensar en algunas analogías que en el contexto de sus estudiantes se puedan utilizar para dar a conocer estos conceptos?

No	Analogía (Nombre)
1	
2	
3	

9. ¿Ha utilizado analogías para el concepto de Ciclo Mientras?

   Si____
   No___

   Si su respuesta es “Si” por favor menciónalas:

No	Analogía (Nombre)
1	
2	
3	

   Si su respuesta es “No” ¿puede por favor pensar en algunas analogías que en el contexto de sus estudiantes se puedan utilizar para dar a conocer estos conceptos?

No	Analogía (Nombre)
1	
2	
3	

10. ¿Ha utilizado analogías para el concepto de Ciclo Hacer Mientras?

Si____
No___

Si su respuesta es “Si” por favor menciónalas:

No	Analogía (Nombre)
1	
2	
3	

Si su respuesta es “No” ¿puede por favor pensar en algunas analogías que en el contexto de sus estudiantes se puedan utilizar para dar a conocer estos conceptos?

No	Analogía (Nombre)
1	
2	
3	

Anexo 5

Formato de Consentimiento Informado

IDIS	**FORMATO** **CONSENTIMIENTO INFORMADO PARA PARTICIPANTES DE INVESTIGACIÓN**	**CÓDIGO:INV-FR-001** **VERSIÓN: 1.0** **FECHA: 01/Febrero/2017**

CONSENTIMIENTO INFORMADO PARA PARTICIPANTES DE INVESTIGACIÓN

El propósito de este documento es la de proveer una clara explicación de la investigación, así como del rol que va a desempeñar Usted como participante.

El estudio está coordinado por _____ identificado con C.C._____.

Es importante resaltar que el principal objetivo de este estudio es "....", y de acuerdo a la información recolectada (propósito de los datos obtenidos).

De igual manera, es importante aclarar que la decisión de participar en el estudio es completamente voluntaria y no tendrá ningún valor monetario y su resultado solo serán empleados con fines académicos. Por último, si tiene dudas sobre el estudio, el equipo de trabajo está disponible para aclararlas.

De antemano, agradecemos su participación y colaboración.

Yo, _____, identificado con C.C._____, acepto mi participación en este estudio coordinado por _____. En este sentido asumo que he sido informado del propósito y del alcance del estudio, por otro lado, reconozco que la información que se obtenga en el estudio es estrictamente confidencial y no será utilizado para ningún otro propósito fuera de los de este estudio sin mi consentimiento.

_____ _____

Nombre del Participante Firma del Participante

_____

Fecha

Anexo 6

Caracterización encuesta

Cod	País	Tipo Universidad		Nivel estudio	Universidad	Docente
		Estatal	Privada			
1	Colombia	X		Doctorado	U. de Nariño	Delio Gómez
2		X		Doctorado	U. de Nariño	Jesús Insuasty
3		X		Maestría	U. de Nariño	Nelson Jaramillo
4		X		Maestría	U. de Nariño	Sandra Vallejo
5		X		Maestría	U. de Nariño	Oscar Revelo
6		X		Maestría	U. de Nariño	Manuel Bolaños
7		X		Especialista	U. de Nariño	Jorge Rivera
8		X		Especialista	U. de Nariño	Oscar Casanova
9			X	Maestría	U. Mariana	Álvaro Martínez
10			X	Maestría	U. Mariana	Andrés Arteaga
11			X	Maestría	U. Mariana	Giovanny Hernández
12			X	Maestría	U. Mariana	Jaime Riascos
13			X	Maestría	U. Mariana	Magda Salazar
14			X	Maestría	U. Mariana	Jaime Salazar
15			X	Maestría	U. CESMAG	Carlos Gonzales
16			X	Maestría	U. CESMAG	Javier Jiménez
17			X	Maestría	U. CESMAG	Joan Ayala
18			X	Maestría	Unicomfacauca	Vanessa Agredo
19		X		Maestría	U. del Cauca	Sandra Buitrón
20			X	Especialista	U. Autónoma	Ginna Leytón
21		X		Maestría	U. Nacional Abierta y a Distancia	Edgar Enríquez
22			X	Maestría	U. Autónoma	Edgar Mauricio Chaves
23			X	Maestría	U. Antonio Nariño	Armando Muñoz
24		X		Doctorado	U. Nacional Abierta y a Distancia	Anivar Chávez
25		X		Maestría	U. Nacional Abierta y a Distancia	Mirian Benavides
26		X		Doctorado	U. Autónoma de Occidente	Andrés Solano
27		X		Maestría	U. del Cauca	Libardo Pantoja
28		X		Maestría	U. San Buenaventura	Betti Grass
29	México	X		Maestría	U. Autónoma de Zacatecas	J. Guadalupe Lara Cisneros
30		X		Doctorado	Tecnológico Nacional de México / IT Aguascalientes	Ricardo Mendoza González
31	Argentina	X		Doctorado	U. Nacional de Santiago del Estero	Rosanna Costaguta
32	España	X		Doctorado	U. Islas Baleares	Cristina Manresa
33	Ecuador	X		Maestría	U. Politécnica Estatal del Carchi	Jorge Miranda Realpe

Anexo 7

Analogías utilizadas para conceptos de Entradas, Salidas, Condicionales y Ciclos.

Entradas	Contexto	Pais	Tipo Universidad	Validadas
Subirse a un bus	Bus	Colombia	Estatal	si
Encendido de un circuito	Circuito	Colombia	Estatal	si
He representado los espacios de la memoria de las computadoras como	Casillero	Colombia	Estatal	si
He utilizado las filas de personas en la sociedad (compra de tickets par	Fila	Colombia	Estatal	si
Escuchar un mensaje y retenerlo en la mente	Mensaje	Colombia	Estatal	si
Combustible de un carro	Carro	Colombia	Estatal	si
Línea de procesos en una fábrica	Fábrica	Colombia	Estatal	si
Baterías de un juguete	Juguete	Colombia	Privada	si
Dato digitado	Dato	Colombia	Privada	si
Ingredientes que se usarán para una comida	Comida	Colombia	Privada	si
Planteamiento de recetas como comparación de requerimientos	Receta	Colombia	Privada	si
Explicación a detalle del entorno educativo para comprensión de actores, clases y atributos	Clase	Colombia	Privada	si
Ingredientes que se usarán para una comida	comida	Colombia	Privada	si
La alimentación, el presupuesto, los diferentes tipos de información cuantitativa, personal humano.	Alimento	Colombia	Privada	si
El pensamiento inicial sobre un problema.	Problema	Colombia	Privada	si
No se puede obtener un resultado de examen cuando todavía no se aplica	Examen	México	Estatal	Si
Licuadora	Licuadora	Argentina	Estatal	Si
Robot ensamblador	Robot	Argentina	Estatal	Si
Poder recibir el teléfono de un contacto, me lo pueden dar por teléfono, por mensaje de whatsapp, por sms, por correo electrónico....	Telefono	España	Estatal	No
Proceso de una fábrica de ensamblaje de vehículos Motor, tornillos, puertas, chasis, etc.	Vehículo	Colombia	Privada	si
Elaboración de una receta de cocina Materiales (huevos, harina, sal, etc)	Cocina	Colombia	Privada	si
El proceso evolutivo (Biológico) del ser humano Células	Humanos	Colombia	Privada	si
Ingreso a un lava autos automático	Lava Autos	Colombia	Privada	si
Ingredientes en receta de cocina	Receta	Colombia	Privada	si
Materiales de construcción	Construcció	Colombia	Privada	si
Valor de un artículo comercial	Precio	Colombia	Privada	si
Consumo de energía eléctrica	Energía	Colombia	Privada	si
Clave en Cajero automático	Cajero	Colombia	Privada	si

Analogías utilizadas para concepto de Salida

Salidas	Contexto	Pais	Tipo Universidad	Validadas
Bajarse del bus	Bus	Colombia	Estatal	si
Un motor comienza a funcionar (salida de circuito)	Motor	Colombia	Estatal	si
Dar información verbal de una situación	Informació	Colombia	Estatal	si
Las facturas de servicios	Factura	Colombia	Estatal	si
Resultado de una operación aritmética	operación	Colombia	Estatal	si
Línea de procesos en una fábrica	fábrica	Colombia	Estatal	si
Funcionamiento de un juguete	Juguete	Colombia	Privada	si
Informe, documento	documento	Colombia	Privada	si
Mensaje informativo	mensaje	Colombia	Privada	si
El plato preparado	Comida	Colombia	Privada	si
Construcción representativa de una solución a partir de un problema dado, haciendo uso de clips	problema	Colombia	Privada	si
Caso especial de comparación de zapato y vestido que cambian de color para contextualizar en la importancia de la interfaz	Vestido	Colombia	Privada	si
El plato preparado	plato	Colombia	Privada	si
El éxito, fin, fracaso, resultado.	situación	Colombia	Privada	si
Cuando fabricamos un producto lo que pretendemos para que tenga éxito y sus ventas aumenten tengo que hacer que sea de calidad.	producto	México	Estatal	Si
Puedo escribir una dirección en un folio, en la mano, en un cuaderno, en un bloc de notas…	informació	España	Estatal	No
Salida del lava autos automático	Lavadero	Colombia	Privada	si
Plato preparado en una receta de cocina	Receta	Colombia	Privada	si
La vivienda en su construcción	vivienda	Colombia	Privada	si
Factura de pago, Recibo de energía.	Factura	Colombia	Privada	si
Plata en Cajero automático	cajero	Colombia	Privada	si
Comida caliente en Horno microondas	Horno	Colombia	Privada	si
Ropa limpia en lavadora	Lavadora	Colombia	Privada	si
La boca y las bocinas estarían relacionadas ya que a través de ellos podemos oír y dar información.	boca	Ecuador	Estatal	si
Sistema renal con impresora proyector auriculares, porque permite ver información que fue procesada por el cuerpo humano y el PC.	Impresora	Ecuador	Estatal	si

Analogías utilizadas para concepto de Condicional Simple

Condicional Simple	Contexto	Pais	Tipo Universidad	Validadas
Para ir a un sitio puede tomar un bus, de lo contrario ir a pie	transporte	Colombia	Estatal	si
Si en un circuito eléctrico uno de los caminos está abierto, la corriente al encontrarse esta situación, obligatoriamente tomará otro camino, (está parcialmente controlada o no controlada, pero se está dando una condición de decisión)	camino	Colombia	Estatal	si
Un paciente que necesita atención médica decide ir o no al médico.	médico	Colombia	Estatal	si
Suelo hablar acerca de las luces automáticas que se activan con la detección de movimiento. Esos mecanismos hacen que las luces permanezcan siempre apagadas como estado natural, y solamente se encienden al detectar el movimiento en un rango cercano.	luz	Colombia	Estatal	si
De igual forma, suelo utilizar el concepto de las puertas eléctricas de un almacén o de un centro comercial. Su estado natural es estar siempre cerradas; éstas solamente se abren al detectar la cercanía de una (o varias) personas.	puerta	Colombia	Estatal	si
Condiciones de situaciones familiares. edad mayor, edad menor.	Edad	Colombia	Estatal	si
Condiciones de situaciones de aula clase: Hay mas mujeres u hombres, el estudiantes mayor, estudiante menor	Género	Colombia	Estatal	si
Condiciones con datos numéricos, par, impar, número mayor y menor, comparación de dos valores.	Par	Colombia	Estatal	si
Condiciones con datos numéricos, par, impar, número mayor y menor, comparación de dos valores.	impar	Colombia	Estatal	si
Condiciones con datos numéricos, par, impar, número mayor y menor, comparación de dos valores.	mayor	Colombia	Estatal	si
Condiciones con datos numéricos, par, impar, número mayor y menor, comparación de dos valores.	menor	Colombia	Estatal	si
Condiciones con situaciones laborales, sueldo mayor, sueldo menor.	sueldo	Colombia	Estatal	si
Escogencia de materiales de construcción de determinado calibre, peso, especificidad en su composición y las razones para seleccionar	materiales	Colombia	Estatal	si
Compra de un producto	compra	Colombia	Estatal	si
Invitación a jugar fútbol dependiendo de las condiciones climáticas	clima	Colombia	Privada	si
Tipos de baterías necesarias para que un juguete funcione	bateria	Colombia	Privada	si
Estados	estado	Colombia	Privada	si
casos	caso	Colombia	Privada	si
Utilizar una sombrilla	sombrilla	Colombia	Privada	si

Analogías utilizadas para concepto de Condicional compuesto

Condicional Compuesto	Contexto	Pais	Tipo Universidad	Validadas
Para ir a un sitio determinado en la misma ciudad, puede ir a pie "o" en algún tipo de transporte.	transporte	Colombia	Estatal	si
Para ir a otra ciudad del país "y" tiene con que pagar, puede ir en bus "o" en avión.	transporte	Colombia	Estatal	si
Una persona con un problema de salud "o" con control periódico puede decidir "ir o no" al médico.	cita médica	Colombia	Estatal	si
Una persona con un problema de salud "y" afiliado a una EPS puede decidir "ir o no" a su EPS.	EPS	Colombia	Estatal	si
En un circuito eléctrico se puede dar esta situación cuando en cada camino hay más de una resistencia las cuales implican un "y".	circuito ele	Colombia	Estatal	si
Cuando las resistencias están en paralelo, se lo puede asimilar a un "o"	resistencia	Colombia	Estatal	si
Suelo hablar acerca de una bifurcación en vía férrea. Dependiendo de "alguna condición", las vías de un tren pueden cambiar en una bifurcación (intercambiador de solo dos caminos posibles) de tal forma que el destino del tren que viaja sobre la vía férrea puede cambiar.	via férrea	Colombia	Estatal	si
Situaciones vida real, condiciones para ir a cine, condiciones para determinar mujeres mayores,	edad	Colombia	Estatal	si
Situaciones con aritmética, tipo de triángulos según lados y ángulos, para esto se colocan ejemplos	triángulo	Colombia	Estatal	si
Situaciones con estudiantes, hombres mayores de cierta edad, hombres o mujeres	mayor edac	Colombia	Estatal	si
Situaciones reales en el campo de la construcción cambiando tamaño del terreno, número de pisos del edificio, número de habitaciones y lograr un total determinado de la obra	construcció	Colombia	Estatal	si
Compra de un producto	producto	Colombia	Estatal	si
Tipos de combos que se puede comprar en el cine dependiendo del dinero que se tenga	cine	Colombia	Privada	si
Tipos de juguetes que pueden funcionar dependiendo del tipo de baterías que requieran	juguetes	Colombia	Privada	si
Alternativas posibles	alternativa	Colombia	Privada	si
Cruzar una calle	calle	Colombia	Privada	si
Vestirse en la mañana	vestuario	Colombia	Privada	si
Preparar un café	café	Colombia	Privada	si

Analogías utilizadas para concepto de Condicional Anidado

Condicional Anidado	Contexto	Pais	Tipo Universidad	Validadas
Si un pasajero está en un paradero de buses, busca que rutas van al sitio que necesita "o" le pregunta a alguna persona si por ahí pasa una ruta que lo lleve donde quiere ir, si la encuentra en la lista o le confirman que si pasa por ahí, tiene que esperar hasta lleque. Si el bus está lleno, tiene que esperar otro.	paradero	Colombia	Estatal	Si
En los circuitos eléctricos se puede dar esta situación cuando se bifurcan los caminos y en cada uno de ellos se instalan resistencias, la corriente en cada bifurcación "tomará la decisión" del camino que menos resistencia le ponga.	circuito	Colombia	Estatal	Si
Si necesita ir a una consulta médica, puede ir a un médico general "o" a un especialista, si necesita un especialista tiene que ver qué tipo de especialista, "si no existe el especialista que requiere, tiene que ir a consulta general y esperar a que lo remitan a otro centro médico.	consulta	Colombia	Estatal	Si
En ocasiones suelo hablar acerca de cómo un equipo de futbol logra en un torneo (por ejemplo el mundial de futbol) ser campeón. Deberá cumplir una serie anidada de condiciones: si el equipo queda de primero (o en ocasiones, de segundo) en la fase de grupos podrá pasar a octavos de final. Si gana el partido en octavos de final, podrá pasar a cuartos de final. Si gana el partido en cuartos de final, pasará a la semifinal. Si gana la semifinal, pasará a la gran final. Y si gana la final se podrá coronar como campeón del torneo.	Futbol	Colombia	Estatal	Si
Hablo acerca del proceso de formación como profesional en el sistema educativo colombiano (en el sector público). Si un individuo desea ser profesional, una serie de condiciones anidadas son requeridas: Primero se debe graduar como bachiller. Si logra ingresar a la universidad pública, deberá aprobar académicamente todos los créditos. Si aprobó académicamente todos los créditos, deberá presentar un trabajo de grado (o su equivalente). Si su trabajo de grado (o equivalente) es aprobado deberá estar a pazysalvo con todas las dependencias de la universidad. Si está a pazysalvo con todas las dependencias de la universidad podrá graduarse como profesional.	Graduarse	Colombia	Estatal	Si
Comparación de dos o más valores	comparació	Colombia	Estatal	Si
Situaciones familiares,	familia	Colombia	Estatal	Si
Situaciones con arimética, determinación de tipo de triángulos,	triángulos	Colombia	Estatal	Si
Situaciones laborales, hombres que ganen más de salario mínimo, cálculo de facturas de servicios con estrato	salario	Colombia	Estatal	Si
Compra de un producto de vestuario	vestido	Colombia	Estatal	Si
Tipos de sitios nocturnos que se pueden visitar dependiendo de la edad y del				

Analogías utilizadas para concepto de Selectiva múltiple

Selectiva Múltiple	Contexto	Pais	Tipo Universidad	Validadas
Si en el paradero de buses pasan varias rutas, el pasajero debe escoger el bus con el identificador de ruta que lo puede llevar al sitio que quiere ir.	buses	Colombia	Estatal	si
Al hacer turno en un Banco, según el tipo de transacción a realizar, se va a la ventanilla correspondiente.	banco	Colombia	Estatal	si
Si un paciente tiene una dolencia o problema de salud y requiere un especialista, acude al consultorio específico y pide una cita.	cita	Colombia	Estatal	si
Retomo la analogía del condicional compuesto, es decir, un intercambiador de vías férreas, donde es posible sumar más caminos de destino.	transporte	Colombia	Estatal	si
Vida diaria. Escoger un plato de un menú de restaurante, Seleccionar bebida en un dispensador de bebidas	comida	Colombia	Estatal	si
Vida laboral: Calculo servicios públicos de acuerdo al estrato	servicios	Colombia	Estatal	si
Máquina surtidora de productos (p.e. bebidas)	máquina	Colombia	Estatal	si
Tipos de sillas que se pueden escoger en un viaje en avión	avión	Colombia	Privada	si
Tipos de cocteles que pueden tomarse dependiendo del licor con que se preparen	coctel	Colombia	Privada	si
Alternativas posibles	alternativa	Colombia	Privada	si
Menú para una semana	comida	Colombia	Privada	si
La ropa a utilizar para una semana	vestuario	Colombia	Privada	si
Clasificar objetos por su forma	clasificar	Colombia	Privada	si
Si usted viene a clases, entonces presta atención, sino, usted hace los ejercicios, de lo contrario usted no está prestando atención	atención	Colombia	Privada	si
Ejemplificaciones con inversiones	inversión	Colombia	Privada	si
Menú de un restaurante	comida	Colombia	Privada	si
N tipos de alternativas.	alternativa	Colombia	Privada	si
Decisiones al mismo tiempo que toma el ser humano	desiciones	Colombia	Privada	si
Si estamos frente a un grupo de personas, y se quiere elegir aquella que su nombre es Pedro Correa	nombre	México	Estatal	No
Se tiene una exposición de vehículos clásico y se tiene que seleccionar aquel que le pertenece.	vehículo	México	Estatal	No
Me dan la indicación de tomar un pastel de una cantidad que se encuentra en un mostrador, pero el que fue colocado primero durante el día.	pastel	México	Estatal	No
En espera de atención en un banco, llamarán al cliente por su número de ficha desde la ventanilla que se desocupe.	banco	México	Estatal	Si
Elegir un combo-desayuno en una cafetería (ofrecen combo 1, combo 2, .. combo n)	combo	Argentina	Estatal	No

Analogías utilizadas para concepto de Ciclo Para

Ciclo Para	Contexto	Pais	Tipo Universidad	Validadas
En un consultorio médico se programan las citas con antelación. Para un día ya se saben cuántos pacientes se van atender, por tanto el proceso de atención se repite por cada paciente (El paciente se ha registrado previamente (entrada); llega el paciente al consultorio a la hora de cita; espera a que lo llamen, el médico lo examina y diagnostica; el paciente sigue las sugerencias del médico y se retira del consultorio (salida)). Esto se **repite** según el número de pacientes de ese	consultorio	Colombia	Estatal	si
El número buses de una ruta para un día están establecidas por la empresa transportadora. Estas rutas hacen el mismo recorrido y proceso (sale de la estación origen, recoge y deja pasajeros, finaliza su recorrido en la estación de destino), esto se repite según el número de buses.	buses	Colombia	Estatal	si
Prefiero mencionar las rutinas de entrenamiento físico de las personas. Hacer exactamente 10 largos en piscina olímpica todos los días. Trotar en un estadio y dar 20 vueltas a la pista. Hacer todos los días 100 sentadillas, etc. La naturaleza de este ciclo es la definición controlada de un rango de iteraciones previamente definidas, esto es, de antemano, se tiene presupuestado el número de iteraciones que el ciclo debe realizar (ni una más y ni una menos).	entrenamie	Colombia	Estatal	si
Secuencia de valores numéricos, sumarlos, contarlos, generar secuencias de números.	aritmética	Colombia	Estatal	si
Vida diaria, sumar las edades de los integrantes de una familia, de un curso.	edades	Colombia	Estatal	si
Trabajo, sumar los salarios de un número dado de empleados.	salarios	Colombia	Estatal	si
Revisar condiciones de la obra con un presupuesto asignado	presupuest	Colombia	Estatal	si
Control de ingreso a un concierto	concierto	Colombia	Estatal	si
500 millas de Indianápolis	indianapol	Colombia	Privada	si
Competencia Keirin en el ciclismo de pista	ciclismo	Colombia	Privada	si
Subir al quinto piso del edificio de laboratorios.	asensor	Colombia	Privada	si
Identificar el número total de compañeras en el curso.	conteo	Colombia	Privada	si
Calcular el promedio de edad de los compañeros y compañeras del curso.	edad	Colombia	Privada	si
Para todas las personas que están en el salón, mencionar el nombre y código	estudiante	Colombia	Privada	si
Juego de las profesiones, en el que se caracteriza cada profesión con las funciones a desempeñar	profesione	Colombia	Privada	si
Analogías por reciprocidad, buscando un fin a un ser.	analogías	Colombia	Privada	si
Para todas las personas que están en el salón, mencionar el nombre y código	estudiante	Colombia	Privada	si
Estudio hasta que termino la carrera.	profesiona	Colombia	Privada	si

Analogías utilizadas para concepto de Ciclo Mientras

Ciclo Mientras	Contexto	Pais	Tipo Universidad	Validadas
Cuando hay varios pasajeros en una parada y NO se sabe la cantidad, cada pasajero (debe ingresar al bus, pagar, marcar, y ubicarse en un puesto si hay disponible de lo contrario debe quedarse parado). El proceso se repite **MIENTRAS** haya pasajeros que necesitan subirse al bus. Se hace un conteo de pasajeros que se suben al bus, mediante la máquina registradora. Esta información es necesaria para futuros procesos.	bus	Colombia	Estatal	si
En un consultorio médico, la recepcionista atiende las solicitudes de los pacientes para programarlos y darles la cita correspondiente. Esto se repite durante todo el día **MIENTRAS** existan usuarios para atenderlos. En el proceso se debe controlar que para un día determinado no se pase de la cantidad que un médico puede atender, además se debe contar la cantidad de pacientes que requieren cita, esto para futuros procesos.	consultorio	Colombia	Estatal	si
En la atención en un banco para realizar transacciones, los usuarios ingresan a la fila o esperan su turno, los cajeros atienden las solicitudes de los usuarios **MIENTRAS** haya clientes.	banco	Colombia	Estatal	si
Los usuarios pueden ingresar **MIENTRAS** estén dentro del horario de atención.	atención	Colombia	Estatal	si
Suelo usar la analogía de la esclusa como en un canal (p.e. el canal de Panamá). Mientras los niveles de agua a ambos lados de la compuerta (esclusa) sean desiguales, esta exclusa permanece cerrada. Solo cuando el nivel del agua es igual, la esclusa se abre para permitir el tránsito de las embarcaciones. La naturaleza del ciclo está en la actividad progresiva del llenado de agua del nivel bajo a un lado de la esclusa. "Mientras que el nivel de agua a los lados de la esclusa sea desigual, la esclusa permanece cerrada"	canal	Colombia	Estatal	si
Vida diaria: sumar edades de los hombres de un curso hasta que llegue una mujer. Contar la cantidad de estudiantes hasta que llegue el primer estudiante con nombre Carlos	conteo	Colombia	Estatal	si
Aritmética: sumar, contar valores numéricos hasta que llegue un valor negativo	aritmética	Colombia	Estatal	si
Trabajo: calcular el valor total pagado por nómina hasta que llegue un trabajador con código 999	nómina	Colombia	Estatal	si
Revisar condiciones de presupuesto hasta llegar al total solicitado	presupuest	Colombia	Estatal	si
Cajero de un supermercado	cajero	Colombia	Estatal	si
Prueba de resistencia en atletismo en un campo de fútbol	atletismo	Colombia	Privada	si
Juego del encuentra la balota ganadora en la bolsa	juego	Colombia	Privada	si
Subir al quinto piso del edificio de laboratorios.	edificio	Colombia	Privada	si

Analogías utilizadas para concepto de Ciclo Hacer Mientras

Ciclo Hacer Mientras	Contexto	Pais	Tipo Universidad	Validadas
El ingreso a un bus se **hace MIENTRAS** se tenga pasajeros y se tenga puestos.	bus	Colombia	Estatal	si
Las consultas que atiende el médico en un día, se **hacen MIENTRAS** se hayan programado para ese día	consultas	Colombia	Estatal	si
El banco atiende usuarios y **hacen** sus transacciones **MIENTRAS** existan clientes en turno.	banco	Colombia	Estatal	si
Retomo el concepto de semáforo. Un semáforo realiza sus cambios de luz temporizadamente (semáforo no inteligente) mientras que se encuentra en el horario de actividad(p.e. entre las 6:00am y las 11:00pm)	semáforo	Colombia	Estatal	si
Revisar condiciones de presupuesto hasta llegar al total solicitado	presupuest	Colombia	Estatal	si
Enjuagado de vehículo al lavarlo	vehículo	Colombia	Estatal	si
Recolectar una cantidad de dinero en el curso.	estudiante	Colombia	Privada	si
Realizar una actividad física de diferentes ejercicios.	ejercicios	Colombia	Privada	si
Preparar una receta de cocina.	receta	Colombia	Privada	si
Asista a todas las clases mientras el semestre dure	estudiante	Colombia	Privada	si
Entregue todas las tareas mientras sean dentro del rango que el profesor propuso	tareas	Colombia	Privada	si
Sume todas las variables mientras la lista tenga datos	lista	Colombia	Privada	si
Enseñanza a partir del Juego de las escondidas	juego	Colombia	Privada	si
Ejemplificación de secuencias hacer mientras haya una condición	actividades	Colombia	Privada	si
Asista a todas las clases mientras el semestre dure	estudiante	Colombia	Privada	si
Entregue todas las tareas mientras sean dentro del rango que el profesor propuso	estudiante	Colombia	Privada	si
Sume todas las variables mientras la lista tenga datos	lista	Colombia	Privada	si
Hasta que piense puedo escribir	pensar	Colombia	Privada	si
Hasta que llueva no debo salir	clima	Colombia	Privada	si
Continuar con el fuego al tazón de agua hasta que el agua esté hirviendo para preparar un rico café	café	México	Estatal	No
Continuar con la entrega de despensas a las familias hasta que se agoten las mismas.	familias	México	Estatal	No
Mientras que el dueño del juego inflable no llegue a decirte que te bajes, tú continúa brincando en el mismo	inflable	México	Estatal	No
En la preparación de limonada, agregar azúcar y menear, probar el agua, continuar agregando azúcar mientras el dulzor no sea el deseado.	limonada	México	Estatal	Si
Jardin (Cortar césped hasta terminar el jardín)	jardín	Argentina	Estatal	Si
Examen (resolver mientras se cumple el tiempo total)	examen	Argentina	Estatal	Si

Anexo 8

Dataset inicial

```
descripcion,accion,verbo,sustantivo,contexto,categoria,pais,tipo_univ,
'subirse a un bus','subir bus',subir,bus,bus,vehiculo,colombia,estatal
'encendido de un circuito','encender circuito',encender,circuito,circu
'he representado los espacios de la memoria de las computadoras como c
'he utilizado las filas de personas en la sociedad (compra de tickets
'escuchar un mensaje y retenerlo en la mente','escuchar mensaje',escuc
'combustible de un carro','llenar combustible',llenar,combustible,carr
'linea de procesos en una fabrica','insertar objeto',insertar,objeto,f
'baterias de un juguete','colocar bateria',colocar,bateria,juguete,jue
'dato digitado','digitar dato',digitar,dato,dato,automatizacion,colomb
'ingredientes que se usaran para una comida','ingresar ingrediente',in
'planteamiento de recetas como comparacion de requerimientos','ingresa
'explicacion a detalle del entorno educativo para comprension de actor
'ingredientes que se usaran para una comida','ingresar ingrediente',in
'la alimentacion el presupuesto los diferentes tipos de informacion cu
'el pensamiento inicial sobre un problema.','pensar inicial',pensar,in
'no se puede obtener un resultado de examen cuando todavia no se aplic
licuadora,'ingresar ingrediente',ingresar,ingrediente,licuadora,comida
'robot ensamblador','ensamblar parte',ensamblar,parte,robot,automatiza
'poder recibir el telefono de un contacto me lo pueden dar por telefon
'proceso de una fabrica de ensamblaje de vehiculos motor tornillos pue
'elaboracion de una receta de cocina materiales (huevos harina sal etc
'el proceso evolutivo (biologico) del ser humano celulas','reproducir
'ingreso a un lava autos automatico','ingresar lava autos',ingresar,la
'ingredientes en receta de cocina','ingresar ingrediente',ingresar,ing
'materiales de construccion','ingresar material',ingresar,material,con
'valor de un articulo comercial','registrar articulo',registrar,articu
'consumo de energia electrica','registrar vatios',registrar,vatios,ene
'clave en cajero automatico','ingresar clave',ingresar,clave,cajero,au
'comida fria en horno microondas','ingresar comida',ingresar,comida,mi
'ropa sucia en lavadora','ingresar ropa',ingresar,ropa,lavadora,automa
'proceso de una fabrica de ensamblaje de vehiculos motor tornillos pue
'elaboracion de una receta de cocina materiales (huevos harina sal etc
'el proceso evolutivo (biologico) del ser humano celulas','reproducir
'el mouse y teclado se los puede comparar con nuestras manos y piernas
```

Anexo 9

Procesamiento con métodos

Método J48

```
=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances         465               81.5789 %
Incorrectly Classified Instances       105               18.4211 %
Kappa statistic                          0.7043
Mean absolute error                      0.1144
Root mean squared error                  0.2391
Relative absolute error                 36.0221 %
Root relative squared error             60.0563 %
Total Number of Instances              570

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,732    0,051    0,612      0,732    0,667      0,630    0,961     0,729     entrada
                 0,477    0,006    0,875      0,477    0,618      0,627    0,973     0,770     salida
                 0,906    0,177    0,828      0,906    0,865      0,730    0,956     0,952     condicional
                 0,789    0,064    0,864      0,789    0,825      0,742    0,965     0,932     ciclos
Weighted Avg.    0,816    0,113    0,823      0,816    0,813      0,716    0,961     0,909

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  41   1   9   5 |   a = entrada
  14  21   7   2 |   b = salida
   7   2 250  17 |   c = condicional
   5   0  36 153 |   d = ciclos
```

Procesamiento método LMT

```
=== Evaluation on training set ===

Time taken to test model on training data: 0.01 seconds

=== Summary ===

Correctly Classified Instances         521               91.4035 %
Incorrectly Classified Instances        49                8.5965 %
Kappa statistic                          0.8633
Mean absolute error                      0.133
Root mean squared error                  0.2119
Relative absolute error                 41.905  %
Root relative squared error             53.2148 %
Total Number of Instances              570

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,839    0,008    0,922      0,839    0,879      0,867    0,995     0,962     entrada
                 0,841    0,002    0,974      0,841    0,902      0,898    0,995     0,966     salida
                 0,931    0,071    0,924      0,931    0,928      0,860    0,976     0,969     condicional
                 0,928    0,061    0,887      0,928    0,907      0,858    0,976     0,951     ciclos
Weighted Avg.    0,914    0,056    0,915      0,914    0,914      0,863    0,979     0,962

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  47   1   5   3 |   a = entrada
   3  37   3   1 |   b = salida
   0   0 257  19 |   c = condicional
   1   0  13 180 |   d = ciclos
```

Procesamiento método Jrip

```
=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances         462               81.0526 %
Incorrectly Classified Instances       108               18.9474 %
Kappa statistic                          0.6818
Mean absolute error                      0.1542
Root mean squared error                  0.2777
Relative absolute error                 48.5705 %
Root relative squared error             69.7366 %
Total Number of Instances              570

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,625    0,014    0,833      0,625    0,714      0,697    0,860     0,618     entrada
                 0,523    0,002    0,958      0,523    0,676      0,692    0,840     0,586     salida
                 0,993    0,323    0,743      0,993    0,850      0,700    0,837     0,742     condicional
                 0,670    0,013    0,963      0,670    0,790      0,732    0,851     0,798     ciclos
Weighted Avg.    0,811    0,162    0,843      0,811    0,803      0,710    0,844     0,737

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  35   0  19   2 |   a = entrada
   0  23  20   1 |   b = salida
   0   0 274   2 |   c = condicional
   7   1  56 130 |   d = ciclos
```

Procesamiento método Modlem

```
=== Evaluation on training set ===

Time taken to test model on training data: 0.02 seconds

=== Summary ===

Correctly Classified Instances         517               90.7018 %
Incorrectly Classified Instances        22                3.8596 %
Kappa statistic                          0.9337
Mean absolute error                      0.0204
Root mean squared error                  0.1429
Relative absolute error                  6.8633 %
Root relative squared error             37.2512 %
UnClassified Instances                  31                5.4386 %
Total Number of Instances              570

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,917    0,002    0,978      0,917    0,946      0,942    0,892     0,789     entrada
                 0,944    0,000    1,000      0,944    0,971      0,970    0,886     0,790     salida
                 0,985    0,058    0,942      0,985    0,963      0,927    0,946     0,917     condicional
                 0,937    0,014    0,973      0,937    0,954      0,931    0,952     0,921     ciclos
Weighted Avg.    0,959    0,034    0,960      0,959    0,959      0,932    0,939     0,899

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  44   0   3   1 |   a = entrada
   1  34   1   0 |   b = salida
   0   0 261   4 |   c = condicional
   0   0  12 178 |   d = ciclos
```

Procesamiento método OneR

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         461               80.8772 %
Incorrectly Classified Instances       109               19.1228 %
Kappa statistic                          0.7106
Mean absolute error                      0.0956
Root mean squared error                  0.3092
Relative absolute error                 30.1137 %
Root relative squared error             77.6586 %
Total Number of Instances              570

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,768    0,146    0,364      0,768    0,494      0,457    0,811     0,303     entrada
                 0,636    0,023    0,700      0,636    0,667      0,641    0,807     0,474     salida
                 0,942    0,024    0,974      0,942    0,958      0,920    0,959     0,945     condicional
                 0,670    0,040    0,897      0,670    0,767      0,686    0,815     0,713     ciclos
Weighted Avg.    0,809    0,041    0,866      0,809    0,825      0,773    0,884     0,767

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  43   2   1  10 |   a = entrada
  10  28   4   2 |   b = salida
   8   5 260   3 |   c = condicional
  57   5   2 130 |   d = ciclos
```

Procesamiento método Part

```
=== Summary ===

Correctly Classified Instances         491               86.1404 %
Incorrectly Classified Instances        79               13.8596 %
Kappa statistic                          0.7843
Mean absolute error                      0.0969
Root mean squared error                  0.2201
Relative absolute error                 30.5167 %
Root relative squared error             55.2769 %
Total Number of Instances              570

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,482    0,008    0,871      0,482    0,621      0,623    0,941     0,648     entrada
                 0,795    0,074    0,473      0,795    0,593      0,573    0,947     0,591     salida
                 0,917    0,007    0,992      0,917    0,953      0,914    0,981     0,978     condicional
                 0,907    0,090    0,838      0,907    0,871      0,802    0,962     0,905     ciclos
Weighted Avg.    0,861    0,041    0,888      0,861    0,865      0,821    0,968     0,891

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  27  17   0  12 |   a = entrada
   0  35   0   9 |   b = salida
   0  10 253  13 |   c = condicional
   4  12   2 176 |   d = ciclos
```

Procesamiento método SMO

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         494               86.6667 %
Incorrectly Classified Instances        76               13.3333 %
Kappa statistic                          0.7881
Mean absolute error                      0.2692
Root mean squared error                  0.341
Relative absolute error                 84.7696 %
Root relative squared error             85.6486 %
Total Number of Instances              570

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.554     0.029      0.674     0.554     0.608      0.821    entrada
                 0.568     0.023      0.676     0.568     0.617      0.817    salida
                 0.942     0.02       0.977     0.942     0.959      0.968    condicional
                 0.918     0.114      0.805     0.918     0.858      0.902    ciclos
Weighted Avg.    0.867     0.053      0.866     0.867     0.864      0.919

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  31   4   1  20 |   a = entrada
   6  25   2  11 |   b = salida
   2   2 260  12 |   c = condicional
   7   6   3 178 |   d = ciclos
```

Procesamiento método InputMappedClassifier

```
=== Summary ===

Correctly Classified Instances         557               97.7193 %
Incorrectly Classified Instances         6                1.0526 %
Kappa statistic                          0.9831
Mean absolute error                      0.0053
Root mean squared error                  0.073
Relative absolute error                  1.705  %
Root relative squared error             18.5078 %
UnClassified Instances                   7                1.2281 %
Total Number of Instances              570

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0,963    0,000    1,000      0,963   0,981      0,979    0,964     0,936     entrada
               1,000    0,000    1,000      1,000   1,000      1,000    0,966     0,937     salida
               0,993    0,007    0,993      0,993   0,993      0,986    0,989     0,985     condicional
               0,990    0,011    0,980      0,990   0,985      0,976    0,990     0,973     ciclos
Weighted Avg.  0,989    0,007    0,989      0,989   0,989      0,983    0,985     0,973

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  52   0   0   2 |   a = entrada
   0  41   0   0 |   b = salida
   0   0 272   2 |   c = condicional
   0   0   2 192 |   d = ciclos
```

Procesamiento método IterativeClassifierOptimizer

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         489               85.7895 %
Incorrectly Classified Instances        81               14.2105 %
Kappa statistic                          0.7696
Mean absolute error                      0.1198
Root mean squared error                  0.2359
Relative absolute error                 37.7261 %
Root relative squared error             59.2537 %
Total Number of Instances              570

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,482    0,014    0,794      0,482    0,600      0,589    0,894     0,595     entrada
               0,409    0,006    0,857      0,409    0,554      0,572    0,914     0,639     salida
               0,938    0,024    0,974      0,938    0,956      0,916    0,969     0,968     condicional
               0,954    0,170    0,743      0,954    0,835      0,748    0,928     0,840     ciclos
Weighted Avg.  0,858    0,071    0,869      0,858    0,849      0,800    0,943     0,862

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  27   2   2  25 |   a = entrada
   1  18   2  23 |   b = salida
   1   0 259  16 |   c = condicional
   5   1   3 185 |   d = ciclos
```

Procesamiento método Apriori

```
Best rules found:

 1. verbo=elegir 51 ==> concepto_general=condicional 51    <conf:(1)> lift:(2.07) lev:(0.05) [26] conv:(26.31)
 2. verbo=jugar 14 ==> concepto_general=ciclos 14    <conf:(1)> lift:(2.94) lev:(0.02) [9] conv:(9.24)
 3. verbo=evaluar sustantivo=edad 12 ==> concepto_general=condicional 12    <conf:(1)> lift:(2.07) lev:(0.01) [6] conv:(6.19)
 4. verbo=jugar contexto=jugar 12 ==> concepto_general=ciclos 12    <conf:(1)> lift:(2.94) lev:(0.01) [7] conv:(7.92)
 5. sustantivo=ingrediente 11 ==> verbo=ingresar 11    <conf:(1)> lift:(18.39) lev:(0.02) [10] conv:(10.4)
 6. verbo=obtener 11 ==> concepto_general=salida 11    <conf:(1)> lift:(12.95) lev:(0.02) [10] conv:(10.15)
 7. verbo=contar 11 ==> concepto_general=ciclos 11    <conf:(1)> lift:(2.94) lev:(0.01) [7] conv:(7.26)
 8. sustantivo=ingrediente 11 ==> concepto_general=entrada 11    <conf:(1)> lift:(10.18) lev:(0.02) [9] conv:(9.92)
 9. sustantivo=ingrediente 11 ==> verbo=ingresar concepto_general=entrada 11    <conf:(1)> lift:(21.11) lev:(0.02) [10] conv:(10.48)
10. verbo=ingresar sustantivo=ingrediente 11 ==> concepto_general=entrada 11    <conf:(1)> lift:(10.18) lev:(0.02) [9] conv:(9.92)
11. sustantivo=ingrediente concepto_general=entrada 11 ==> verbo=ingresar 11    <conf:(1)> lift:(18.39) lev:(0.02) [10] conv:(10.4)
12. sustantivo=multiplos 8 ==> verbo=hacer 8    <conf:(1)> lift:(16.29) lev:(0.01) [7] conv:(7.51)
13. contexto=multiplos 8 ==> verbo=hacer 8    <conf:(1)> lift:(16.29) lev:(0.01) [7] conv:(7.51)
14. sustantivo=operacion 8 ==> concepto_general=condicional 8    <conf:(1)> lift:(2.07) lev:(0.01) [4] conv:(4.13)
15. sustantivo=multiplos 8 ==> contexto=multiplos 8    <conf:(1)> lift:(71.25) lev:(0.01) [7] conv:(7.89)
16. contexto=multiplos 8 ==> sustantivo=multiplos 8    <conf:(1)> lift:(71.25) lev:(0.01) [7] conv:(7.89)
17. sustantivo=multiplos 8 ==> concepto_general=ciclos 8    <conf:(1)> lift:(2.94) lev:(0.01) [5] conv:(5.28)
18. contexto=multiplos 8 ==> concepto_general=ciclos 8    <conf:(1)> lift:(2.94) lev:(0.01) [5] conv:(5.28)
19. sustantivo=multiplos 8 ==> verbo=hacer contexto=multiplos 8    <conf:(1)> lift:(71.25) lev:(0.01) [7] conv:(7.89)
20. verbo=hacer sustantivo=multiplos 8 ==> contexto=multiplos 8    <conf:(1)> lift:(71.25) lev:(0.01) [7] conv:(7.89)
21. contexto=multiplos 8 ==> verbo=hacer sustantivo=multiplos 8    <conf:(1)> lift:(71.25) lev:(0.01) [7] conv:(7.89)
22. verbo=hacer contexto=multiplos 8 ==> sustantivo=multiplos 8    <conf:(1)> lift:(71.25) lev:(0.01) [7] conv:(7.89)
23. sustantivo=multiplos contexto=multiplos 8 ==> verbo=hacer 8    <conf:(1)> lift:(16.29) lev:(0.01) [7] conv:(7.51)
24. sustantivo=multiplos 8 ==> verbo=hacer concepto_general=ciclos 8    <conf:(1)> lift:(17.27) lev:(0.01) [7] conv:(7.54)
25. verbo=hacer sustantivo=multiplos 8 ==> concepto_general=ciclos 8    <conf:(1)> lift:(2.94) lev:(0.01) [5] conv:(5.28)
26. sustantivo=multiplos concepto_general=ciclos 8 ==> verbo=hacer 8    <conf:(1)> lift:(16.29) lev:(0.01) [7] conv:(7.51)
27. contexto=multiplos 8 ==> verbo=hacer concepto_general=ciclos 8    <conf:(1)> lift:(17.27) lev:(0.01) [7] conv:(7.54)
28. verbo=hacer contexto=multiplos 8 ==> concepto_general=ciclos 8    <conf:(1)> lift:(2.94) lev:(0.01) [5] conv:(5.28)
29. contexto=multiplos concepto_general=ciclos 8 ==> verbo=hacer 8    <conf:(1)> lift:(16.29) lev:(0.01) [7] conv:(7.51)
30. verbo=evaluar sustantivo=clima 8 ==> concepto_general=condicional 8    <conf:(1)> lift:(2.07) lev:(0.01) [4] conv:(4.13)
31. verbo=evaluar sustantivo=numeros 8 ==> concepto_general=condicional 8    <conf:(1)> lift:(2.07) lev:(0.01) [4] conv:(4.13)
32. sustantivo=numeros concepto_general=condicional 8 ==> verbo=evaluar 8    <conf:(1)> lift:(2.95) lev:(0.01) [5] conv:(5.29)
```

Procesamiento método PredictiveApriori

Best rules found:

1. concepto=ciclo_hacer_mientras 128 ==> concepto_general=ciclos 128 acc:(0.9949)
2. verbo=evaluar 193 ==> concepto_general=condicional 192 acc:(0.99486)
3. concepto=cond_simple 76 ==> concepto_general=condicional 76 acc:(0.9947)
4. concepto=cond_completo 72 ==> concepto_general=condicional 72 acc:(0.99467)
5. concepto=ciclo_para 66 ==> concepto_general=ciclos 66 acc:(0.9946)
6. concepto=cond_anidado 64 ==> concepto_general=condicional 64 acc:(0.99458)
7. concepto=select_multiple 64 ==> concepto_general=condicional 64 acc:(0.99458)
8. concepto=entrada 56 ==> concepto_general=entrada 56 acc:(0.99445)
9. concepto_general=entrada 56 ==> concepto=entrada 56 acc:(0.99445)
10. verbo=elegir 51 ==> concepto_general=condicional 51 acc:(0.99434)
11. concepto=salida 44 ==> concepto_general=salida 44 acc:(0.99412)
12. concepto_general=salida 44 ==> concepto=salida 44 acc:(0.99412)
13. categoria=calculo concepto=cond_completo 20 ==> verbo=evaluar concepto_general=condicional 20 acc:(0.99113)
14. verbo=jugar 14 ==> concepto_general=ciclos 14 acc:(0.98772)
15. verbo=hacer categoria=calculo 14 ==> concepto_general=ciclos 14 acc:(0.98772)
16. verbo=ensamblar 12 ==> categoria=automatizacion 12 acc:(0.98554)
17. verbo=ingresar categoria=comida 12 ==> concepto=entrada concepto_general=entrada 12 acc:(0.98554)
18. verbo=obtener 11 ==> concepto=salida concepto_general=salida 11 acc:(0.98407)
19. verbo=contar 11 ==> concepto_general=ciclos 11 acc:(0.98407)
20. sustantivo=ingrediente 11 ==> verbo=ingresar categoria=comida 11 acc:(0.98407)
21. sustantivo=ingrediente 11 ==> verbo=ingresar concepto=entrada 11 acc:(0.98407)
22. verbo=ingresar sustantivo=ingrediente 11 ==> concepto_general=entrada 11 acc:(0.98407)
23. verbo=jugar concepto=ciclo_hacer_mientras 10 ==> contexto=jugar 10 acc:(0.98223)
24. verbo=calcular 9 ==> categoria=calculo 9 acc:(0.97988)
25. contexto=calculadora concepto_general=condicional 9 ==> categoria=automatizacion 9 acc:(0.97988)
26. sustantivo=operacion 8 ==> categoria=automatizacion concepto_general=condicional 8 acc:(0.97683)
27. verbo=hacer sustantivo=multiplos 8 ==> contexto=multiplos 8 acc:(0.97683)
28. verbo=generar 7 ==> categoria=automatizacion concepto=salida 7 acc:(0.97277)
29. verbo=generar 7 ==> categoria=automatizacion concepto_general=salida 7 acc:(0.97277)
30. verbo=sumar 6 ==> categoria=calculo concepto_general=ciclos 6 acc:(0.96716)
31. sustantivo=parte 6 ==> categoria=automatizacion 6 acc:(0.96716)
32. sustantivo=transporte 6 ==> concepto_general=condicional 6 acc:(0.96716)
33. sustantivo=camino 6 ==> concepto_general=condicional 6 acc:(0.96716)
34. sustantivo=nota 6 ==> verbo=evaluar concepto_general=condicional 6 acc:(0.96716)

Procesamiento método EM

```
EM
==

Number of clusters selected by cross validation: 4
Number of iterations performed: 0
```

Attribute	Cluster 0 (0.41)	1 (0.4)	2 (0.08)	3 (0.1)
verbo				
subir	4	1	1	2
encender	2	2	1	2
llenar	5	1	1	3
hacer	36	1	1	1
escuchar	4	1	1	2
insertar	1	1	1	2
colocar	1	1	1	2
digitar	1	1	1	2
ingresar	5	1	1	28
explicar	1	1	1	2
introducir	1	1	1	2
pensar	1	1	1	2
aplicar	1	1	1	2
ensamblar	8	3	1	4
registrar	4	1	1	5
reproducir	1	1	1	3
realizar	6	1	1	2
observar	1	1	1	2
leer	1	1	1	3
coger	1	1	1	2
entrar	1	1	1	2
iniciar	2	1	1	2
salir	2	1	1	1

Anexo 10

Etiquetación morfosintáctica

mod → (T) Modifier	**det** → (T) Nominal Specifier (or determiner)	**aux** → (T) Specifier
dobj → (T) Direct object	**iobj** → (T) Indirect object	**pobj** → ?
mwe → (T) Verb particle	**cc** → (T) Coordination	**punct** → (T) Punctuation markup
case → (T) Clitic relation	**cop** → (T) Verbal predicative	**circ** → (T) Prepositional object of a verb
nmod → (T) Prepositional object of a noun	**subj** → (T) Subject	

(T) Head position	(T) Head	(T) Head category	(T) Dependent position	(T) Dependent	(T) Dependent category	(T) Syntactic relation
0	ingrediente	nombre	2	receta	nombre	Complemento nominal
2	receta	nombre	4	cocina	nombre	Complemento nominal

mod → (T) Modifier	**det** → (T) Nominal Specifier (or determiner)	**aux** → (T) Specifier
dobj → (T) Direct object	**iobj** → (T) Indirect object	**pobj** → ?
mwe → (T) Verb particle	**cc** → (T) Coordination	**punct** → (T) Punctuation markup
case → (T) Clitic relation	**cop** → (T) Verbal predicative	**circ** → (T) Prepositional object of a verb
nmod → (T) Prepositional object of a noun	**subj** → (T) Subject	

(T) Head position	(T) Head	(T) Head category	(T) Dependent position	(T) Dependent	(T) Dependent category	(T) Syntactic relation
3	articular	verbo	4	comercial	adjetivo	Atributo verbal

mod → (T) Modifier	**det** → (T) Nominal Specifier (or determiner)	**aux** → (T) Specifier
dobj → (T) Direct object	**iobj** → (T) Indirect object	**pobj** → ?
mwe → (T) Verb particle	**cc** → (T) Coordination	**punct** → (T) Punctuation markup
case → (T) Clitic relation	**cop** → (T) Verbal predicative	**circ** → (T) Prepositional object of a verb
nmod → (T) Prepositional object of a noun	**subj** → (T) Subject	

(T) Head position	(T) Head	(T) Head category	(T) Dependent position	(T) Dependent	(T) Dependent category	(T) Syntactic relation
0	comida	nombre	1	frío	adjetivo	Modificador
0	comida	nombre	3	horno	nombre	Complemento nominal
3	horno	nombre	4	microondas	nombre	Modificador

Anexo 11

Datos enlazados con Tripletas

perimental

Supercorrector

Extractor de tripletas

Sujeto	Relacion	Objeto
baterias de un juguete dato digitado ingredientes	`se usaran para`	una comida planteamiento de recetas
casillas	`representan`	una direccion de memoria
su contenido	`equivale a`	el valor
todavia	`cuando no se aplica`	poder recibir el telefono de un contacto
la harina	`requiere`	la tarjeta de ahorro que
la harina	`requiere la tarjeta de ahorro que para`	hacer el pan
que	`requiere`	juego de el
que	`requiere juego de el para`	comprar algo
que	`requiere juego de el de`	codigo de barras la disposicion de bandejas
producto final	`como es`	incorporar los ingredientes necesarios para poder obtener el producto lectura en voz alta entrada
lectura boca	`preparacion de`	un alimento
dispensador de bebidas con dinero	`entra`	dinero
ingredientes	`preparacion de`	un alimento – producto

Anexo 12

Libro Metacognición

La metacognición y la teoría de la actividad en la enseñanza de la programación

Primera edición, octubre 2017

AUXILIARES DE INVESTIGACIÓN
Midred Natalia Cuasialpud Alvarado
James Montilla
Mónica Natali Palacios Bastidas
Pablo Portilla

Bajo el Sello Editorial CESMAG
Carrera 20A No.14-54
San Juan de Pasto, Nariño, Colombia

Romero Chaves, Cristina.
La metacognición y la teoría de la actividad en la enseñanza de la programación/ Cristina Romero Chaves, María Mercedes Rosero Sosa, Javier Jiménez Toledo. -- 1 ed. -- San Juan de Pasto: Institución Universitaria Centro de Estudios Superiores María Goretti, 2017.

234 p. : il. ; 14 cm.

Incluye Referencias Bibliográfica p. 216 – 233

ISBN: 978-958-56354-3-2
e-ISBN: 978-958-56354-4-9
DOI: 10.15658/CESMAG16.010302

1. METACOGNICION 2. ENSEÑANZA DE LA PROGRAMACION 3. PEDAGOGIA DE LA INGENIERIA DE SISTEMAS 4. EDUCACION EN LA INGENIERIA

CDD 796.077
20. Ed.

CEP - Institución Universitaria Centro de Estudios Superiores María Goretti CESMAG. Biblioteca Remigio Fiore Fortezza.

Editores:
Diego Martínez Hernández
Emma del Pilar Rojas Vergara

Edición impresa y digital

Anexo 13

Libro Pensamiento Computacional

Propuesta metodológica para el desarrollo del pensamiento computacional en la formación de Contadores Públicos desde el componente informático mediante hojas de cálculo.
Primera edición, 2020

Editorial Universidad Cesmag
Carrera 20A # 14-54
Tel: +572 – 7216535 ext: 377 – 218
E-mail: editorial@unicesmag.edu.co
Website: www.unicesmag.edu.co
San Juan de Pasto, Nariño, Colombia
CP: 520003

E-mail: dxramos@unicesmag.edu.co
E-mail: jajimenez@unicesmag.edu.co
San Juan de Pasto, Nariño, Colombia
CP: 520003

ISBN: 978-958-5504-58-5
e-ISBN: 978-958-5504-59-2
DOI: 10.15658/CESMAG20.02180114

Rector
Daniel Omar Sarria Tejada OFM. Cap.

Director editorial
Javier Alejandro Jiménez Toledo

Edición
Diana Milena Betancourth Castillo

Impreso y hecho en Colombia
Printed and made in Colombia

Diseño de cubierta y diagramación
D.G. Daniel Portilla - danielportillaf@hotmail.com

APA:

Ramos, D. y Jiménez, J. (2020). *Propuesta metodológica para el desarrollo del pensamiento computacional en la formación de Contadores Públicos desde el componente informático mediante hojas de cálculo.* Pasto, Colombia: Editorial universidad Cesmag. DOI: 10.15658/CESMAG20.02180114

Anexo 14

AlgoritBuild

ALGORITBUILD: APLICATIVO PARA POTENCIAR EL DESARROLLO DEL PENSAMIENTO ALGORITMICO EN ESTUDIANTES DE PRIMER CURSO DE PROGRAMACION

ERIKA DANIELA REYES CUARAN
SAIDY FERNANDA TUMAL GUZMAN

ASESOR:
PhD. (c). JAVIER JIEMENEZ TOLEDO

INSTITUCIÓN UNIVERSITARIA CESMAG
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERIA DE SISTEMAS
SAN JUAN DE PASTO
2019

Anexo 15

Plataforma desarrollo de Pensamiento Algorítmico

PLATAFORMA BASADA EN ANALOGÍAS PARA EL DESARROLLO DEL PENSAMIENTO ALGORÍTMICO

MENESES BOTINA WILMER ANDRES

Informe final para optar el título de Ingeniero de Sistemas.

Director:
PhD. Javier Jiménez Toledo

UNIVERSIDAD CESMAG
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS

Anexo 16

AlgoVirtual

REALIDAD VIRTUAL PARA EL DESARROLLO DE PENSAMIENTO ALGORİTMICO EN DOCENTES UNIVERSITARIOS, CASO UNIVERSIDAD CESMAG

AUTORES

DANIEL ESTEBAN MADROÑERO MUÑOZ

CHRISTIAN DANIEL GOYES MUÑOZ

Informe final como requisito para optar al título de ingeniera de sistemas

ASESOR

PhD. (c) JAVIER ALEJANDRO JIMÉNEZ TOLEDO

FACULTAD DE INGENIERİA

PROGRAMA DE INGENIERİA DE SISTEMAS

Anexo 17

Director Proyecto de Maestría Unicauca

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones

TESIS DE POSGRADO

FORMATO I: **ACTA DE SUSTENTACIÓN**

TESIS DE: ■ **Maestría** □ **Doctorado**

LOS JURADOS DE LA TESIS TITULADA:
Técnica de conformación de grupos en escenarios de aprendizaje colaborativo basada en rasgos de la personalidad para la enseñanza de la programación

HACEN CONSTAR:

Que siendo las 8:30 a.m. del día 20 del mes de Junio de 2018 realizó la sustentación de tesis el estudiante: Oscar Revelo Sánchez dirigido por: Javier Jiménez Toledo y César Collazos Ordoñez

OBTENIENDO EL CONCEPTO DE: ■ **APROBADO** □ **APROBADO CON OBSERVACIONES**

□ **APLAZADO** □ **NO APROBADO**

Para constancia se firma en Popayán, a los 20 días del mes de junio de 2018.

Jurado Coordinador:
Nombre: Beatriz Eugenia Grass

Jurado 1:
Nombre: Beatriz Eugenia Grass

Jurado 2:
Nombre: Miguel Angel Redonco

Jurado 3:
Nombre:

Con anuencia de los Jurados, firma el Coordinador del Programa,

Nombre: Carlos Cobos

Acuerdo 41/89, recibo de tesorería No.

Anexo 18

Admisión Héctor Mora a Doctorado Unicauca

Universidad del Cauca
Vigilada Mineducación

Sistema de Inscripciones
Programas de Posgrados

- NOTIFICACIÓN DE ADMISIÓN -

Apreciado aspirante, queremos informarle que has sido admitido al programa **Doctorado en Ciencias de la Electrónica**.

Por favor ingrese al aplicativo de inscripción en donde podrá consultar los conceptos de pago para realizar su matrícula financiera.

Bienvenido a la Universidad del Cauca.

Anexo 19

Admisión Joan Ayala a Doctorado Unicauca

Inscripciones Posgrados Unicauca <simcaposnoti02@unicauca.edu.co>
para

Universidad del Cauca

Sistema de Inscripciones
Programas de Posgrados

- NOTIFICACIÓN DE ADMISIÓN -

Apreciado aspirante, queremos informarle que has sido admitido al programa **Doctorado en Ciencias de la Electrónica**.

Por favor ingrese al aplicativo de inscripción en donde podrá consultar los conceptos de pago para realizar su matrícula financiera.

Bienvenido a la Universidad del Cauca.

Servicio notificación Inscripciones Posgrado

Por favor no responda a este correo electrónico.

Anexo 20

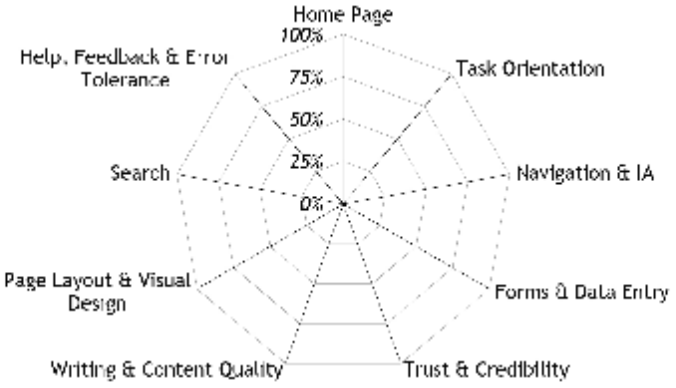
Expert Review Checkpoint de Travis

USERFOCUS

Expert Review

Summary of results

	Raw score	# Questions	# Answers	Score
Home Page	0	20	0	
Task Orientation	0	44	0	
Navigation & IA	0	29	0	
Forms & Data Entry	0	23	0	
Trust & Credibility	0	13	0	
Writing & Content Quality	0	23	0	
Page Layout & Visual Design	0	38	0	
Search	0	20	0	
Help, Feedback & Error Tolerance	0	37	0	
Overall score		247	0	



Home Page

Checkpoint

The items on the home page are clearly focused on users' key tasks ("featuritis" has been avoided)	☐
The home page contains a search input box	☐
Product categories are provided and clearly visible on the homepage	☐
Useful content is presented on the home page or within one click of the home page	☐
The home page shows good examples of real site content	☐
Links on the home page begin with the most important keyword (e.g. "Sun holidays" not "Holidays in the sun")	☐
There is a short list of items recently featured on the homepage, supplemented with a link to archival content	☐
Navigation areas on the home page are not over-formatted and users will not mistake them for adverts	☐
The value proposition is clearly stated on the home page (e.g. with a tagline or welcome blurb)	☐
The home page contains meaningful graphics, not clip art or pictures of models	☐
Navigation choices are ordered in the most logical or task-oriented manner (with the less important components information at the bottom)	☐
The title of the home page will provide good visibility in search engines like Google	☐
All corporate information is grouped in one distinct area (e.g. "About Us")	☐
Users will understand the value proposition	☐
By just looking at the home page, the first time user will understand where to start	☐
The home page shows all the major options	☐
The home page of the site has a memorable URL	☐
The home page is professionally designed and will create a positive first impression	☐
The design of the home page will encourage people to explore the site	☐
The home page looks like a home page; pages lower in the site will not be confused with it	☐

Task Orientation & Site Functionality

Checkpoint	
The site is free from irrelevant, unnecessary and distracting information	
Excessive use of scripts, applets, movies, audio files, graphics and images has been avoided	
The site avoids unnecessary registration	
The critical path (e.g. purchase, subscription) is clear, with no distractions on route	
Information is presented in a simple, natural and logical order	
The number of screens required per task has been minimised	
The site requires minimal scrolling and clicking	
The site correctly anticipates and prompts for the user's probable next activity	
When graphs are shown, users have access to the actual data (e.g. numeric annotation on bar charts)	
Activities allocated to the user or the computer take full advantage of the strengths of each (look for actions that can be done automatically by the site, e.g. postcode lookup)	
Users can complete common tasks quickly	
Items can be compared easily when this is necessary for the task (e.g. product comparisons)	
The task sequence parallels the user's work processes	
The site makes the user's work easier and quicker than without the system	
The most important and frequently used topics, features and functions are close to the centre of the page, not in the far left or right margins	
The user does not need to enter the same information more than once	
Important, frequently needed topics and tasks are close to the 'surface' of the web site	
Typing (e.g. during purchase) is kept to an absolute minimum, with accelerators ("one-click") for return users	
The path for any given task is a reasonable length (2-5 clicks)	
When there are multiple steps in a task, the site displays all the steps that need to be completed and provides feedback on the user's current position in the workflow	
Price is always clearly displayed next to any product	
The site has a prominent, distinctive, consistent, and descriptive label that tells users what information and services are provided	

Navigation & Information Architecture

Checkpoint	
There is a convenient and obvious way to move between related pages and sections and it is easy to return to the home page	
The information that users are most likely to need is easy to navigate to from most pages	
Navigation choices are ordered in the most logical or task-oriented manner	
The navigation system is broad and shallow (many items on a menu) rather than deep (many many levels)	
The site structure is simple, with a clear conceptual model and no unnecessary levels	
The major sections of the site are available from every page (persistent navigation) and there are no dead ends	
Navigation tabs are located at the top of the page, and look like clickable versions of real-world tabs	
There is a site map that provides an overview of the site's content	
The site map is linked to from every page	
The site map provides a concise overview of the site, not a rehash of the main navigation or a list of every single topic	
Good navigational feedback is provided (e.g. showing where you are in the site)	
Category labels accurately describe the information in the category	
Links and navigation labels contain the "trigger words" that users will look for to achieve their goal	
Terminology and conventions (such as link colours) are (approximately) consistent with general web usage	
Links look the same in the different sections of the site	
Product pages contain links to similar and complementary products to support cross-selling	
The terms used for navigation items and hypertext links are unambiguous and jargon-free	

Forms & Data Entry

Checkpoint

Fields in data entry screens contain default values when appropriate and show the structure of the data and the field length	☐
When a task involves source documents (such as a paper form), the interface is compatible with the characteristics of the source document	☐
The site automatically enters field formatting data (e.g. currency symbols, commas for 1000s, trailing or leading spaces). Users do not need to enter characters like $ or %	☐
Field labels on forms clearly explain what entries are desired	☐
Text boxes on forms are the right length for the expected answer	☐
There is a clear distinction between "required" and "optional" fields on forms	☐
The same form is used for both logging in and registering (i.e. it's like Amazon)	☐
Forms pre-warn the user if external information is needed for completion (e.g. a passport number)	☐
Questions on forms are grouped logically, and each group has a heading	☐
Fields on forms contain hints, examples or model answers to demonstrate the expected input	☐
When field labels on forms take the form of questions, the questions are stated in clear, simple language	☐
Pull-down menus, radio buttons and check boxes are used in preference to text entry fields on forms (i.e. text entry fields are not overused)	☐
With data entry screens, the cursor is placed where the input is needed	☐
Data formats are clearly indicated for input (e.g. dates) and output (e.g. units of values).	☐

Trust & Credibility

Checkpoint

The content is up-to-date, authoritative and trustworthy	☐
The site contains third-party support (e.g. citations, testimonials) to verify the accuracy of information	☐
It is clear that there is a real organisation behind the site (e.g. there is a physical address or a photo of the office)	☐
The company comprises acknowledged experts (look for credentials)	☐
The site avoids advertisements, especially pop-ups.	☐
Delivery costs are highlighted at the very beginning of checkout	☐
The site avoids marketing waffle	☐
Each page is clearly branded so that the user knows he is still in the same site	☐
It is easy to contact someone for assistance and a reply is received quickly	☐
The content is fresh: it is updated frequently and the site includes recent content	☐
The site is free of typographic errors and spelling mistakes	☐
The visual design complements the brand and any offline marketing messages	☐
There are real people behind the organisation and they are honest and trustworthy (look for bios)	☐

Writing & Content Quality

Checkpoint	
The site has compelling and unique content	☐
Text is concise, with no needless instructions or welcome notes	☐
Each content page begins with conclsuions or implications and the text is written with an inverted pyramid style	☐
Pages use bulleted and numbered lists in preference to narrative text	☐
Lists are prefaced with a concise introduction (e.g. a word or phrase), helping users appreciate how the items are related to one another	☐
The most important items in a list are placed at the top	☐
Information is organised hierarchically, from the general to the specific, and the organisation is clear and logical	☐
Content has been specifically created for the web (web pages do not comprise repurposed material from print publications such as brochures)	☐
Product pages contain the detail necessary to make a purchase, and users can zoom in on product images	☐
Hypertext has been appropriately used to structure content	☐
Sentences are written in the active voice	☐
Pages are quick to scan, with ample headings and sub-headings and short paragraphs	☐
The site uses maps, diagrams, graphs, flow charts and other visuals in preference to wordy blocks of text	☐
Each page is clearly labelled with a descriptive and useful title that makes sense as a bookmark	☐
Links and link titles are descriptive and predictive, and there are no "Click here!" links	☐
The site avoids cute, clever, or cryptic headings	☐
Link names match the title of destination pages, so users will know when they have reached the intended page	☐
Button labels and link labels start with action words	☐
Headings and sub-headings are short, straightforward and descriptive	☐
The words, phrases and concepts used will be familiar to the typical user	☐
Numbered lists start at "1" not at "0"	☐
Acronyms and abbreviations are defined when first used	☐

Page Layout & Visual Design

Checkpoint	
The screen density is appropriate for the target users and their tasks	☐
The layout helps focus attention on what to do next	☐
On all pages, the most important information (such as frequently used topics, features and functions) is presented on the first screenful of information ("above the fold")	☐
The site can be used without scrolling horizontally	☐
Things that are clickable (like buttons) are obviously pressable	☐
Items that aren't clickable do not have characteristics that suggest that they are	☐
The functionality of buttons and controls is obvious from their labels or from their design	☐
Clickable images include redundant text labels (i.e. there is no 'mystery meat' navigation)	☐
Hypertext links are easy to identify without needing to 'minesweep' (e.g. underlined)	☐
Fonts are used consistently	☐
The relationship between controls and their actions is obvious	☐
Icons and graphics are standard and/or intuitive (concrete and familiar)	☐
There is a clear visual "starting point" to every page	☐
Each page on the site shares a consistent layout	☐
Pages on the site are formatted for printing, or there is a printer-friendly version	☐
Buttons and links show that they have been clicked	☐
GUI components (like radio buttons and check boxes) are used appropriately	☐
Fonts are readable	☐
The site avoids italicised text and uses underlining only for hypertext links	☐
There is a good balance between information density and use of white space	☐
The site is pleasant to look at	☐
Pages are free of "scroll stoppers" (headings or page elements that create the illusion that users have reached the top or bottom of a page when they have not)	☐
The site avoids extensive use of upper case text	☐
The site has a consistent, clearly recognisable look and feel that will engage users	☐
Saturated blue is avoided for fine detail (e.g. text, thin lines and symbols)	☐
Colour is used to structure and group items on the page	☐
Graphics will not be confused with banner ads	☐

Search

Checkpoint	
The default search is intuitive to configure (no Boolean operators)	
The search results page shows the user what was searched for and it is easy to edit and resubmit the search	
Search results are clear, useful and ranked by relevance	
The search results page makes it clear how many results were retrieved, and the number of results per page can be configured by the user	
If no results are returned, the system offers ideas or options for improving the query based on identifiable problems with the user's input	
The search engine handles empty queries gracefully	
The most common queries (as reflected in the site log) produce useful results	
The search engine includes templates, examples or hints on how to use it effectively	
The site includes a more powerful search interface available to help users refine their searches (preferably named "power search" or "refine search", not "advanced search")	
The search results page does not show duplicate results (either perceived duplicates or actual duplicates)	
The search box is long enough to handle common query lengths	
Searches cover the entire web site, not a portion of it	
If the site allows users to set up a complex search, these searches can be saved and executed on a regular basis (so users can keep up to date with dynamic content)	
The search interface is located where users will expect to find it (top right of page)	
The search box and its controls are clearly labelled (multiple search boxes can be confusing)	
The site supports people who want to browse and people who want to search	
The scope of the search is made explicit on the search results page and users can restrict the scope (if relevant to the task)	
The search results page displays useful meta-information, such as the size of the document, the date that the document was created and the file type (Word, pdf etc.)	
The search engine provides automatic spell checking and looks for plurals and synonyms	
The search engine provides an option for similarity search ("more like this")	

Checkpoint	
The FAQ or on-line help provides step-by-step instructions to help users carry out the most	
It is easy to get help in the right form and at the right time	
Prompts are brief and unambiguous	
The user does not need to consult user manuals or other external information to use the site	
The site uses a customised 404 page, which includes tips on how to find the missing page and	
The site provides good feedback (e.g. progress indicators or messages) when needed (e.g.	
Users are given help in choosing products	
User confirmation is required before carrying out potentially "dangerous" actions (e.g. deleting	
Confirmation pages are clear	
Error messages contain clear instructions on what to do next	
Immediately prior to commiting to the purchase, the site shows the user a clear summary page	
When the user needs to choose between different options (such as in a dialog box), the options	
The site keeps users informed about unavoidable delays in the site's response time (e.g. when	
Error messages are written in a non-derisory tone and do not blame the user for the error	
Pages load quickly (5 seconds or less)	
The site provides immediate feedback on user input or actions	
The user is warned about large, slow-loading pages (e.g. "Please wait..."), and the most	
Where tooltips are used, they provide useful additional help and do not simply duplicate text in	
When giving instructions, pages tell users what to do rather than what to avoid doing	
The site shows users how to do common tasks where appropriate (e.g. with demonstrations of the	
The site provides feedback (e.g. "Did you know?") that helps the user learn how to use the site	
The site provides context sensitive help	
Help is clear and direct and simply expressed in plain English, free from jargon and buzzwords	
The site provides clear feedback when a task has been completed successfully	