Como citar este artículo: J. A. Jiménez Toledo, C. A. Collazos, M. Ortega and D. X. Ramos, "Algorithmic Thinking and Extension of its Definition for Trainee Software Developers: A Systematic Literature Mapping," in *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 18, no. 4, pp. 331-343, Nov. 2023, doi: 10.1109/RITA.2023.3323784.

Pensamiento Algorítmico y extensión de su definición para Desarrolladores de Software en Formación: un Mapeo Sistemático de Literatura

Javier Alejandro Jiménez Toledo, César A. Collazos, Manuel Ortega and Deixy Ximena Ramos

***Title*—Algorithmic thinking and extension of its definition for training software developers: a systematic literature mapping**

***Abstract*— This paper exhibits a systematic literature mapping of the considerations required to develop algorithmic thinking in a first course in computer programming (CS1) in university academic programs in computing. In the methodological process of this study, 5 stages were proposed: research questions, search, selection, quality assessment and synthesis extraction. In this way, 5 guiding questions were drawn, 136 articles generated by the search stage were analyzed and the synthesis of 55 documents that met the criteria of this research was concluded, thus compiling the different practices used for the development of algorithmic thinking. In addition, as a result of the systematic mapping of literature, a definition of Algorithmic Thinking oriented Software Engineering and didactics is proposed.**

***Index Terms*— Algorithms, computation, computational logic, computational thinking, computer programming.**

I. INTRODUCCIÓN

El pensamiento algorítmico disciplina, ordena y formaliza el pensamiento del estudiante [1] y su estudio en edades tempranas conlleva la preparación requerida en programas universitarios o tecnológicos como en áreas ingenieriles, matemáticas, físicas, contables, entre otras [2], además, quienes lo estudian poseen varias competencias metacognitivas, creativas, de orientación y de pensamiento [3].

El desarrollar pensamiento algorítmico es un reto y para muchos estudiantes el obstáculo en un primer curso de programación, donde la apropiación y correcta utilización de la estructura de control y selección es lo más complejo [4]. 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 [5].

El pensamiento algorítmico en este contexto implica no solo la capacidad de escribir código, sino también de analizar y comprender problemas, diseñar algoritmos efectivos y depurar errores. Es por ello, que en el ámbito de la programación, esta habilidad se vuelve aún más relevante, ya que permite a los estudiantes enfrentar desafíos de codificación de manera estructurada y efectiva [6].

Este estudio exhibe un mapeo sistemático de literatura de las consideraciones requeridas para desarrollar pensamiento algorítmico en un primer curso de programación de computadores (CS1) en programas académicos universitarios en computación a través de 5 etapas: preguntas de investigación, búsqueda, selección, evaluación de calidad y extracción y síntesis. Finalmente se trabajó con 55 estudios, presentando al final una definición de pensamiento algorítmico desde un marco orientado por la Ingeniería de Software y la didáctica.

Este artículo se encuentra estructurado de la siguiente manera: la Sección 1 presenta una descripción preliminar sobre la relevancia y los retos del pensamiento algorítmico; la Sección 2 desarrolla el mapeo sistemático de literatura utilizado en este estudio; la Sección 3 muestra los hallazgos de dicho mapeo y presenta una extensión para la definición de pensamiento algorítmico en un CS1; la sección 4 incluye las conclusiones; y finalmente se registran los referentes citados en este documento.

.

II. METODOLOGÍA

El mapeo sistemático de literatura utilizado en este estudio tomó como base el enfoque científico de la Ingeniería de Software [7][8][9][10][11] con el propósito de facilitar una amplia perspectiva de una temática en particular, registrando los estudios encontrados y clasificándolos según criterios establecidos para elaborar procesos y estructuras de categorización válidas para una comunidad científica. En este sentido, este enfoque ofrece un conjunto de principios y técnicas rigurosas que garantizan la validez y confiabilidad de los resultados obtenidos en el mapeo sistemático de literatura. Así mismo al utilizar este enfoque, se siguieron pasos metodológicos bien definidos, como la formulación de preguntas de investigación, la selección y análisis de las fuentes de información pertinentes, y la síntesis de los hallazgos encontrados [12]. Dicho enfoque tiene una alta aceptación porque construye conocimiento a partir de las divulgaciones generadas en el desarrollo de actividades investigativas [13]. Las etapas para el mapeo sistemático de literatura del presente estudio se muestran en la Fig. 1.

Fig. 1. Etapas del mapeo sistemático

A. Preguntas de investigación

La revisión sistemática de literatura propuesta en este estudio, inició con la formulación de 5 preguntas mediante las cuales se identificó los procesos claves para el desarrollo de pensamiento algorítmico en un CS1, estas fueron:

RQ1: ¿Cuáles son los problemas que se presentan al afrontar un CS1?

RQ2: ¿Qué reflexiones se encuentra en la literatura sobre algoritmos en un CS1?
RQ3: ¿Cómo se concibe el pensamiento computacional en un CS1?
RQ4: ¿Cuáles son las consideraciones que hacen los autores sobre el pensamiento algorítmico en un CS1?
RQ5: ¿Qué aspectos didácticos existen en torno al pensamiento algorítmico en un CS1?

Para la formulación de las preguntas se utilizó PIPOH del estándar PICO [14], mediante los criterios presentados en la Tabla I.

TABLA I. CRITERIOS ESTABLECIDOS CON PIPOH

Criterio	Detalle
Población	Desarrolladores de software en formación
Intervención	Problemas de enseñanza/aprendizaje, definiciones, didácticas, estrategias
Resultados	Documentos validados por una comunidad académico-científica relacionados con el desarrollo de pensamiento algorítmico
Profesionales	Computación y áreas afines
Contexto	Educación universitaria

B. Búsqueda

Se definió como estrategia de búsqueda la exploración en bases bibliográficas del término clave principal, el cual fue complementado y combinado mediante términos equivalentes (ver Tabla II) para obtener una amplia variedad de documentos a procesar.

TABLA II. ELEMENTOS DE BÚSQUEDA

Término principal	Término equivalente	Complementos
Algorithmic Thinking	Algorithmic skills Learning or teaching in computer programming Teaching environments for computer programming First course in computer programming CS1 Teaching-learning strategies in programming	Education Computer or Engineering Publication >= 2015

En la Tabla III se presentan las bases de datos bibliográficas consideradas en este estudio, las cuales fueron elegidas por su importancia científica y a la vez por la cantidad de documentos que la comunidad académico-científica de las ciencias computacionales tienen publicados en estas bases de datos.

TABLA III. BASES DE DATOS BIBLIOGRÁFICAS UTILIZADAS

Base de datos	Idioma del documento
Scopus	Inglés
DOAJ	Inglés
Redalyc	Español
IEEE	Inglés

Posteriormente se construye la cadena de búsqueda mediante la combinación del término principal con los términos equivalentes y los complementos a través de operadores lógicos. Dicha cadena fue: (“Algorithmic thinking” OR “algorithmic skills” OR “Learning or teaching in computer programming” OR “Teaching environments for computer programming” OR “First course in computer programming” OR “CS1” OR “Teaching-learning strategies in programming”) AND (education Or Computer Or Engineering) AND (publication>= 2015).

C. Selección

Se analizó todos los documentos obtenidos al aplicar la cadena de búsqueda en las bases de datos bibliográficas mencionadas anteriormente, para ello se consideraron los siguientes elementos: título, resumen, palabras claves, introducción, estudios relacionados, marcos teóricos o conceptuales, método o metodología, resultados, discusión, conclusiones y trabajos futuros.

Además, en esta etapa se implementó criterios de inclusión y exclusión acordes a las preguntas de investigación y que fijaron condiciones de calidad en el proceso de selección, dichos criterios se mencionan a continuación:

- ✓ Artículos de investigación sobre pensamiento algorítmico en un CS1 en programas profesionales universitarios.
- ✓ Documentos con investigaciones terminadas
- ✓ Versiones completas de publicaciones.

Por su parte, los criterios de exclusión establecieron:

- ✓ Investigaciones que no se encuentren en las ciencias de la computación o sus equivalentes.
- ✓ Documentos no relacionados con procesos educativos.
- ✓ Archivos duplicados.
- ✓ Documentos en idiomas distintos al inglés, portugués o español.
- ✓ Documentos no disponibles para descarga

Una vez definidos las pautas exclusión e inclusión, se llevaron a cabo dos actividades importantes para en el proceso de selección:

1. Revisión de título, resumen y conclusiones. Permitió identificar los documentos duplicados y no relevantes para el trabajo.
2. Selección de documentos de calidad. Se realizó teniendo en cuenta los criterios de evaluación de calidad.

D. Evaluación de calidad

Esta etapa se realiza en la tarea de selección de documentos mencionada anteriormente, para ello se aplicaron siete procesos: origen de los documentos, importancia del documento, incidencia de la investigación, objetivo de estudio, ámbito de aplicación, validez del método y rigor científico del estudio. Dichos procesos se enmarcan en los 3 elementos que contempla la gestión de calidad: organización, planificación y control.

E. Extracción de datos y síntesis de resultados

Una vez concluida la etapa evaluación de calidad se procede a la extracción de los datos teniendo en cuantas las 5 preguntas de investigación planteadas en este estudio que tuvo como propósito el caracterizar los procesos de desarrollo de pensamiento algorítmico llevados a cabo en un CS1. Concluida la extracción de información se realiza el proceso de síntesis, cuyos resultados se presentan en la sección 3.

III. RESULTADOS

En esta sección se presentan los hallazgos del mapeo sistemático de literatura, compilando las consideraciones publicadas en la literatura científica para el desarrollo de pensamiento algorítmico en un CS1. La duración de este

estudio fue de 5 meses comprendidos desde enero hasta mayo de 2023.

Teniendo en cuenta que se consultó cuatro bases de datos bibliográficas, los procesos de búsqueda, selección y evaluación de calidad se realizaron por separado en cada una de ellas, como se presenta en la Tabla IV

TABLA IV. TIEMPOS DE BÚSQUEDA, SELECCIÓN Y EVALUACIÓN DE CALIDAD

B/D Bibliográfica	Periodo
Scopus	25/01/2023
DOAJ	13/02/2023
Redalyc	03/03/2023
IEEE	09/04/2023

La aplicación de la cadena de búsqueda en las bases de datos bibliográficas de la tabla 5 generó un total de 136 estudios que al aplicarles el proceso de selección y calidad se encontraron 16 archivos duplicados y se excluyeron 67 documentos, obteniéndose así un total de 55 estudios pertinentes como lo muestra la Tabla V.

TABLA V. RESULTADOS DE ETAPA DE SELECCIÓN Y EVALUACIÓN DE CALIDAD

Docs. encontrados	Docs. duplicados	Docs. Descartados	Docs. Pertinentes	B/D Bibliográfica
20	2	3	15	Scopus
22	1	5	16	DOAJ
41	9	24	8	Redalyc
53	4	35	16	IEEE
136	16	67	55	

Además, los años 2015, 2016 y 2018 fue donde se encontró la mayor cantidad de estudios y cuyo detalle total de los 55 documentos contemplados en este mapeo sistemático se muestra en la Tabla VI.

TABLA VI. CITAS POR AÑO

Año	Archivos	Citas bibliográficas
2015	10	[5][15][16][17][18][19][20][21][22][23]
2016	12	[4][14][24][25][26][27][28][29][30][31][32][33]
2017	5	[34][35][36][37][38]
2018	11	[1][2][3][13][39][40][41][42][43][44][39]
2019	5	[45][46][47][48][105]
2020	2	[86][92]
2021	1	[53]
2022	5	[60][70][71][85][88]
2023	4	[6][12][102][106]
Total	55	

Para complementar los datos presentados en la tabla 6, se realizó una caracterización por base de datos y año de los 55 estudios seleccionados observando la frecuencia de los mismos en el tiempo (ver Tabla VII).

TABLA VII. ESTUDIOS POR BASE DE DATOS Y AÑO DE PUBLICACIÓN

BD	2015	2016	2017	2018	2019	2020	2021	2022	2023	Total
Scopus	5	3	2	2	0	1	0	1	1	15
DOAJ	2	3	2	5	1	0	0	2	1	16
Redalyc	2	3	0	3	0	0	0	0	0	8
IEEE	1	3	1	1	4	1	1	2	2	16
Total	10	12	5	11	5	2	1	5	4	55

Además, en esta revisión se encontraron dos tipos de documentos: artículos científicos publicados en revistas y contribuciones de eventos investigativos. Es así como de los 55 documentos obtenidos, 44 corresponden a artículos y 10 son contribuciones, siendo Scopus, DOAJ e IEEE las bases de datos con mayor número de artículos y Redalyc quien aporta mayor número de actas como se muestra en la fig. 2.

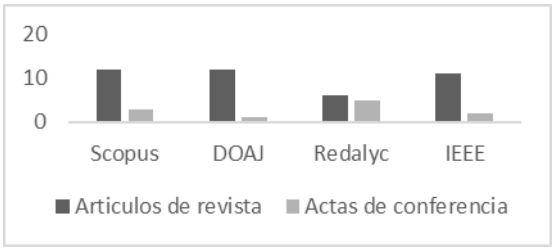

Fig. 2. Documentos por tipo de publicación

A continuación, se describen los hallazgos del mapeo sistémico de literatura, los cuales se encuentran organizados en seis secciones, de las cuales, las cinco primeras son acordes con las preguntas formuladas en la sección II.A y una sección adicional donde se propone una definición de pensamiento algorítmico desde la Ingeniería de Software, dichas secciones son las siguientes: Problemas en los procesos de enseñanza/aprendizaje en un CS1, consideraciones sobre algoritmos, importancia del pensamiento computacional, características del pensamiento algorítmico, consideraciones didácticas del pensamiento algorítmico y pensamiento algorítmico para desarrolladores de software.

A. Problemas en los procesos de enseñanza/aprendizaje en un CS1.

El desarrollar pensamiento algorítmico en futuros programadores se inicia en el CS1 recibido, el cual es un desafío para docentes como para estudiantes novatos debido a su complejidad [49], por ello, muchos profesionales informáticos en unión con otros de áreas como la educación y Psicología tiene la misma preocupación [34] que ha conllevado a la realización de estudios en procesos de enseñanza y también de aprendizaje que plantean posibles soluciones para mejorar los resultados obtenidos [16][24]. A pesar de ello, hoy en día no se cuenta con soluciones consensuadas para afrontar los diversos problemas existentes en los dos procesos antes mencionados [17][50][51] [52].

Los cursos de programación hoy en día presentan una serie de desafíos en los procesos de enseñanza/aprendizaje debido a su naturaleza técnica, entre los problemas más frecuentes se encuentran la dificultad para comprender conceptos abstractos, la carencia de una buena práctica y la ausencia de motivación constructiva [53], es por ello, que las estadísticas de los dos últimos decenios demuestran la pérdida de interés de los estudiantes en programas profesionales relacionados con las ciencias de la computación [54][52] a pesar de la afinidad que hoy se tiene con dispositivos tecnológicos [52] y por si fuera poco, son muchas las dificultades de aprendizaje reportadas en la literatura científica al abordar un CS1 donde el estudiante presenta inconvenientes de motivación, conocimientos previos, adaptabilidad, métodos de estudio [51][55][56][57] [58], interpretación errónea de enunciados, y aunque varios

alcanzan a terminan dicho primer curso, aún poseen cantidad de falencias [24] derivados de asimilar en tan poco tiempo habilidades de resolución de problemas con quizá nuevos modelos mentales basados en supuestos matemáticos[24][25][59].

Por lo anterior, se identificó que la falta de ejercicios prácticos y la retroalimentación limitada en las tareas asignadas pueden impedir el desarrollo de habilidades de programación sólidas [60], por consiguiente es necesario el uso de estrategias pedagógicas efectivas que fomenten un aprendizaje más activo, práctico y personalizado en los cursos de programación

Es necesario que un estudiante cuente con determinadas habilidades de pensamiento antes de comenzar un CS1 [24], porque debe asimilar en poco tiempo un conjunto de conocimientos complejos [61], que muchos de ellos al finalizarlo no alcanzan a desarrollar las metas propuestas, asumiendo un postura de rechazo [62][63][64][24] que lo conlleva a posibles frustraciones que pueden terminar en deserción académica [25][65][66].

Uno de los problemas más recurrentes en los noveles estudiantes son los relacionados con procesos de abstracción [67] que afectan directamente la apropiación de lógica computacional, que finalmente perjudica los resultados obtenidos en el diseño algorítmico [68] debido a la complejidad exigida en el proceso cognitivo propios del pensamiento de orden superior [59][24] al tratar de asimilar conceptos como variables, memoria, tipado de datos, entre otros [69], en el contexto de la educación, los estudiantes noveles a menudo enfrentan desafíos significativos en los procesos de abstracción, los cuales implican la capacidad de comprender y aplicar conceptos abstractos en situaciones concretas [70], así mismo, la falta de esta habilidad para la abstracción puede complicar su capacidad para razonar y resolver problemas de manera efectiva o transferir sus conocimientos a nuevas situaciones [71].

Asimismo, enfrentarse a una estructura sintáctica y semántica en un códigos fuentes y que a pesar que en la actualidad existen una amplia cantidad y variedad de recursos didácticos disponibles, estos no cuentan con la calidad necesaria para tal fin [24][72].

Como si lo anterior no fuera poco, el paradigma de programación elegido para un CS1 también es tema de discusión, involucrando el Orientada a Objetos (POO), Estructurado, Orientado a Eventos e incluso el Lógico o Funcional [73], donde el POO a pesar de ser uno de los mayormente utilizados, su aprendizaje requiere elementos adicionales como el dominio de su propio entorno de desarrollo, metodologías propias, manejo de patrones y la constante actualización de sus versiones enfocadas al ámbito profesional antes que el académico[18].

Así como existen problemas relacionados directamente desde la perspectiva del estudiante, también los profesores desde su ámbito de enseñanza de la programación de computadores tiene sus inconvenientes para desarrollar pensamiento algorítmico en el estudiantado, dichas dificultades inician por decidir si comenzar con un enfoque «object-first» o «procedural-first» [74], además, las inapropiadas metodologías de enseñanza puede ser un factor crítico [68] [73] que a pesar de la actual revolución educativa, aún contemplan aspectos didácticos de la vieja escuela donde predominan los modelos por repetición que no favorecen las necesidades actuales sobre todo en este campo [75].

El desarrollo de pensamiento algorítmico asociado directamente con los proceso de enseñanza tiene su principal preocupación en el accionar didáctico del profesor en un CS1 [76], al no propiciar desde un método de estudio adecuado como no contemplar elementos de planeación, control y evaluación apropiados para este tipo de aprendizaje [77], basándose en metodologías que obligan al estudiante a realizar una amplia cantidad de ejercicios mediante prueba-error que finalmente terminan en un proceso poco productivo [78]. Es por ello, que la utilización de recursos inapropiados, como materiales desactualizados o recursos que no se tengan en cuenta sus estilos de aprendizaje individuales puede afectar su compromiso y participación en el aula llegando así a tener un impacto negativo en el proceso de enseñanza-aprendizaje [79]. Asimismo, la utilización de recursos de enseñanza inapropiados agrava más la situación en el afán proveer al estudiante en herramientas para mejorar la solucionar problemas, aprender uno o más lenguajes de programación [73], etc.

Otros inconvenientes reportados al afrontar un CS1 son: ambigüedad en los enunciados presentados[67], falta de ejercicios de creatividad computacional [19], innovación en los métodos de enseñanza [19][39][80], niveles de complejidad erróneos en los ejercicios planteados [26], planteamiento de ejercicios que causan confusión y desmotivación [45], ejemplos demasiado fáciles, abordar excesivos conceptos, incluir temas irrelevantes, dedicación de trabajo solo con los estudiantes aventajados [40], alta carga cognitiva [81], tareas mal construidas[28], elección de lenguajes de programación para profesionales y no para principiantes [82], métodos de evaluación algorítmicos inadecuados [41], entre otros.

B. Consideraciones sobre algoritmos

En los estudios realizados por Knuth [74] 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 [27]. 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 [27].

Futschek propone el algoritmo como un método de resolución de problemas mediante instrucciones debidamente determinadas [83] donde no existe un algoritmo para diseñar algoritmos [42]. Además, un algoritmo es la colección de reglas, que no están sujetas a otras reglas [38], para realizar cálculos en forma manual o computarizada [84], 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 [19] e incluyen

métodos de modelación y creatividad y no dependen de los lenguajes de programación ya que no necesariamente se deben computarizar [27] y se pueden manifestar como procedimientos matemáticos, líneas de código o datos y pueden expresarse de diferentes formas [39], es fundamental que los algoritmos sean comprensibles y puedan justificar las decisiones tomadas, lo que contribuye a la confianza y la aceptación de su implementación. Asimismo, debe ser preciso y tener un orden lógico de ejecución, cada paso debe estar claramente definido, sin ambigüedades, y seguir un flujo lógico que garantice la correcta resolución del problema [85].

Por otro lado, un algoritmo es una representación formal y un enfoque creativo [1]. A su vez, los algoritmos de tipo informático concentran elementos invisibles y al mismo tiempo son abstractos en su estructura al ser diseñados para un entorno computacional [80] [26]. Todo algoritmo debe cumplir cinco condiciones: finitud, definibilidad, entradas, salidas y efectividad [74].

Según la Real Academia Española [45], la palabra algoritmo proviene posiblemente del latín "algobarismus" como abreviación del cálculo de cifras arábigas y hace referencia a "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".

C. Importancia del Pensamiento computacional

El pensamiento computacional es una destreza necesaria para cualquier usuarios en el siglo XXI [40] y propone un paradigma de enseñanza-aprendizaje con características cuantitativas, centrado en la intuición y que incorpora conocimiento mediante prueba y error [81] 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. [28] [82] [41]. 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 problémicas en diversas áreas [86], aplicando conceptos básicos informáticos [43].

Wing 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 [87].

Hoy en día, la enseñanza de pensamiento computacional ha tomado gran importancia en escuelas primarias en todo el mundo [29], incorporándola de forma transversal en cursos como matemáticas, ciencias, arte y ciencias sociales, sin aumentar las horas de clase [46]. Es así, como la enseñanza del pensamiento computacional es esencial en la educación actual. Proporciona a los estudiantes habilidades fundamentales como el pensamiento crítico, la resolución de problemas, la creatividad y la comprensión del mundo digital. Estas habilidades son valiosas en múltiples contextos y preparan a los estudiantes para enfrentar los desafíos del siglo XXI de manera efectiva [88]

D. Características del Pensamiento algorítmico

El pensamiento algorítmico tiene relación directa con el pensamiento computacional [89] [35][40] y es una habilidad que todas las personas deben tener [19], además, es una competencia que incorpora el pensamiento abstracto y lógico, el pensamiento en estructuras, la creatividad, etc., para resolver problemas [90], permitiendo plantear una solución mediante una serie de pasos [19], logrando representar situaciones complejas mediante instrucciones simples [91].

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 [23].

Asimismo, el pensamiento algorítmico es un sistema de métodos mentales [92]y permite que el estudiante desarrolle su pensamiento, analice un contexto y diseñe una solución adecuada sin utilizar un lenguaje de programación [20], de esta manera el pensamiento algorítmico puede producir conocimiento conceptual [15] debido a que es una habilidad matemática mediante la cual es posible solucionar problemas, modelarlos algorítmicamente y automatizarlos [3]. 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 [93], es decir, proporciona un procedimiento para resolver un problema [40].

El pensamiento algorítmico estructura el pensar de forma lógica y organizada y mediante herramientas divide una situación compleja en una serie de acciones simples [30], además, provee la capacidad de ejecutar, validar, entender y construir algoritmos para afrontar diversas situaciones [94].

Analoca 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) [27].

El pensamiento algorítmico también se concibe como una habilidad para construir, concebir, ejecutar y evaluar procesos computacionales [40] [21], entendiendo y ejecutando acciones paso a paso y que permitan la creación de nuevos algoritmos [19] con procesos de razonamiento y lógica en la escritura de las instrucciones requeridas para solucionar un problema [95]

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 [43], 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 [1].

Sarienko[1] describe que la base del pensamiento algorítmico está conformada por las cinco etapas de formación espiral de acciones mentales de Halperin [96]: 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 [97] para el desarrollo de pensamiento algorítmico mediante el cual el estudiante realiza etapas de planificación, ejecución y control en su proceso de aprendizaje [36].

Por su parte, Landa 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 [98].

Asimismo, Semenihina y Rudenko [2] 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 [47] 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 [31] 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 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 [32].

Al igual, Altukhova y Smirnova [32] 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 [33] 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 [83] 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.

Así mismo, el pensamiento algorítmico tiene la capacidad de dividir un problema complejo en pasos más pequeños y manejables, establecer patrones y reglas, y a la vez utilizar la lógica deductiva para tomar decisiones, fomentando la abstracción para así simplificar un problema centrándose en aspectos esenciales e ignorar detalles irrelevantes [88].

Finalmente, Christodoulou afirma que el pensamiento algorítmico no se refiere únicamente 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 [42].

E. Consideraciones didácticas del pensamiento algorítmico.

En los procesos de aprendizaje, la motivación es esencial para iniciar y catalizar las habilidades requeridas en el desarrollo del pensamiento algorítmico [99] y una de las estrategias de aprendizaje más utilizadas es el reconocimiento de patrones, el cual realiza acciones como escuchar, ver y hacer [19].

Por su parte, la enseñanza del pensamiento algorítmico trae consigo un conjunto de problemas didácticos, lo cual lo hace verdaderamente relevante [1], incluyendo una serie de consideraciones y principios que los estudiantes deben desarrollar como motivación continua, participación activa, reto progresivo y abstracción [19]. Además, es necesario desarrollas estructuras mentales como: secuencia, selección y repetición mediante pseudocódigo, diagramas de flujo y lenguaje de programación [27].

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 [37]. Computacionalmente existen diversas herramientas para desarrollo de pensamiento algorítmico, entre ellas: pseudocódigo, bloques visuales, diagramas de flujo, diagramas de actividad, diagrama NS, etc. [29]

Csernoch, Biró, Máth y Abari 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 [5].

En la enseñanza de la lógica algorítmica, la herramienta de mayor aceptación la constituyen los visualizadores de algoritmos [83] los cuales permiten “visualizar” uno o varios seguimientos por cada instrucción a través de representaciones gráficas y/o utilizando 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 [100]. Además, en la enseñanza del pensamiento algorítmico se debe desarrollar tres habilidades: pensamiento creativo, raciocinio sistémico y trabajo colaborativo [101];

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 proponen el modelo ADRI como un enfoque de enseñanza para la generación de pensamiento algorítmico que combina elementos de enfoque, despliegue, resultado y mejora [47].

Así mismo, la enseñanza práctica juega un papel muy importante en el pensamiento algorítmico, apoyando a los estudiantes ha aprender de una mejor manera a través de la práctica activa, proporcionando numerosas oportunidades para resolver problemas algorítmicos, incluyendo ejercicios, desafíos, proyectos y simulaciones [102].

A su vez, M. Zhaldak y Kurilov propone incorporar métodos de enseñanza activos para el desarrollo de pensamiento algorítmico como el aula invertida, el ABP y el Aprendizaje Colaborativo y Cooperativo [103][22]; mientras que Minty [104] 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 [2].

Por otro lado, Lamagna [21] propone un enfoque desenchufado 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 [92] y Castelblanco et al. [48], plantean el desarrollo del pensamiento algorítmico por medio de aplicaciones robóticas; Mezak y Papak [44] diseñan escenarios de aprendizaje adecuados con situaciones cotidianas; Solomon[105] aborda la fuerza de la representaciones espaciales como elemento central para la enseñanza y el aprendizaje del pensamiento algorítmico en el aula de clase; Vinayakumar et, al. [40] utiliza la programación visual basada en bloques.

F. Pensamiento Algorítmico para Desarrolladores de Software

Los desarrolladores con un sólido pensamiento algorítmico tienen la facultad de analizar problemas, identificar patrones y diseñar soluciones eficientes, así mismo, tienen la habilidad para crear aplicaciones y sistemas de software más robustos y eficientes [106].

De acuerdo con el mapeo sistemático de literatura y teniendo en cuenta la experiencia de los autores, en esta sección se propone una definición de pensamiento algorítmico desde un marco orientado por la Ingeniería de Software y la didáctica para programadores de computadores.

Para ello, se propone que la generación 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 profesores y estudiantes tienen un rol fundamental. En la Figura 3 se presenta una aproximación del modelo para desarrollo de pensamiento algorítmico de acuerdo a la definición propuesta.

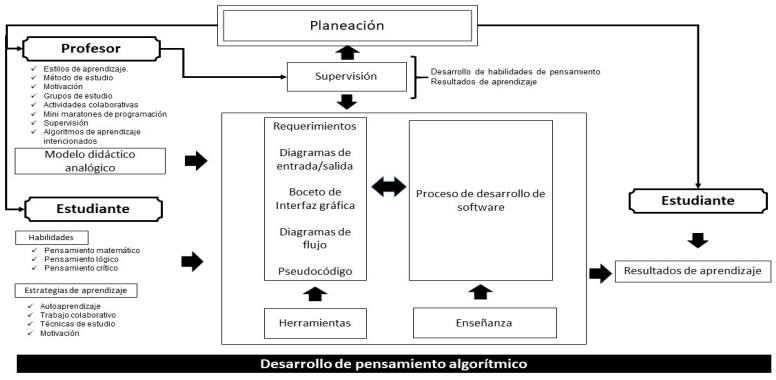

Fig. 3. Modelo desarrollo de pensamiento algorítmico

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, porque, desde este nivel, el estudiante debe comprender que el desarrollo de software obedece a una colección de actividades planificadas y bien diferenciadas.

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.

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.

Los estudiantes que afrontan un CS1 deben tener las siguientes habilidades para desarrollar un nivel inicial de pensamiento algorítmico:

Pensamiento matemático estable. Ya que en la mayoría de casos es la única habilidad formalmente desarrollada 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.

Asimismo, 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 habilidad 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 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 habilidades 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 no inhabilita 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 importante que el profesor identifique el estilo de aprendizaje mayormente utilizado por los estudiantes, 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 docente 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 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.

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.

Además, en este modelo se propone hacer evidente la utilización del modelo didáctico analógico 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 durante el CS1 y así obtener los mejores resultados de aprendizaje en el estudiante.

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 4 se detalla el proceso de enseñanza/aprendizaje en relación con las herramientas requeridas para la generación de pensamiento algorítmico.

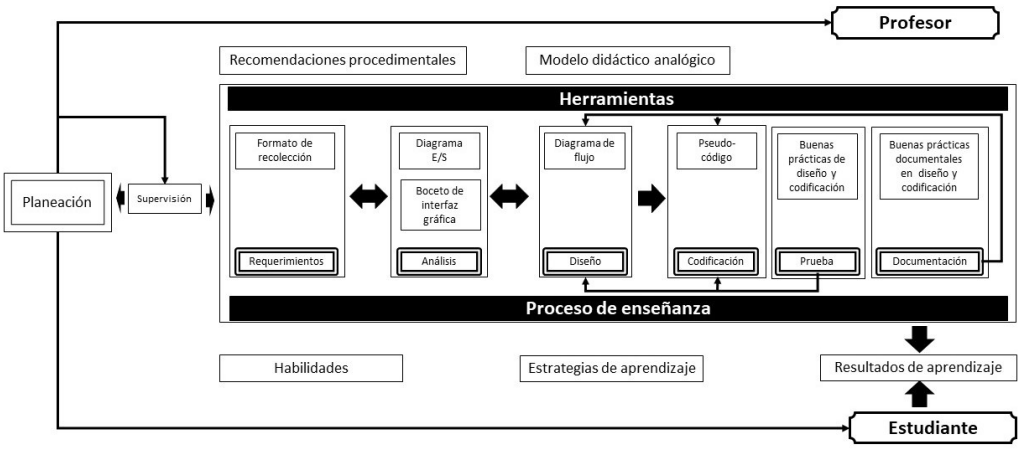

Fig. 4. Modelo enseñanza para desarrollo de pensamiento algorítmico

Además, es necesario que el docente 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 docente y cuyo formato se encuentra en la tabla VIII.

TABLA VIII. FORMATO PARA RECOLECTAR REQUERIMIENTOS

Nombre del sistema	Acción
Requerimientos de entrada	- Capturar... - Capturar…
Requerimientos de salida	- Imprimir… - Imprimir…

Análisis. Se requiere que el profesor le indique al estudiante que posterior a la fase de recolección de requerimientos, se realizará el análisis de la información, cuyo objetivo es determinar mediante la simulación de un entorno Cliente – Desarrollador de software (donde el docente asumirá el rol de cliente y el estudiante de desarrollador de software), la claridad que debe tener el desarrollador de software frente a las necesidades de un cliente para que el proyecto 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 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 (Graphic Interface Sketch).

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 fig. 5).

Fig. 5. BlackBox.

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.

Graphic Interface Sketch (GIS). En esta fase se concibe un primer boceto general 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 GIS permite representar los elementos analizados en un Diagrama de E/S. Además, el GIS 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 fig. 6).

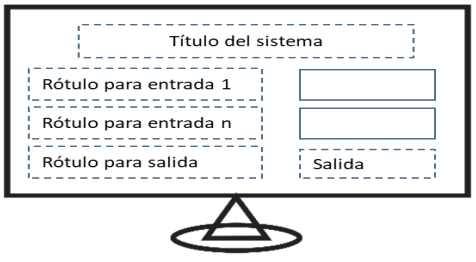

Fig.6. GIS

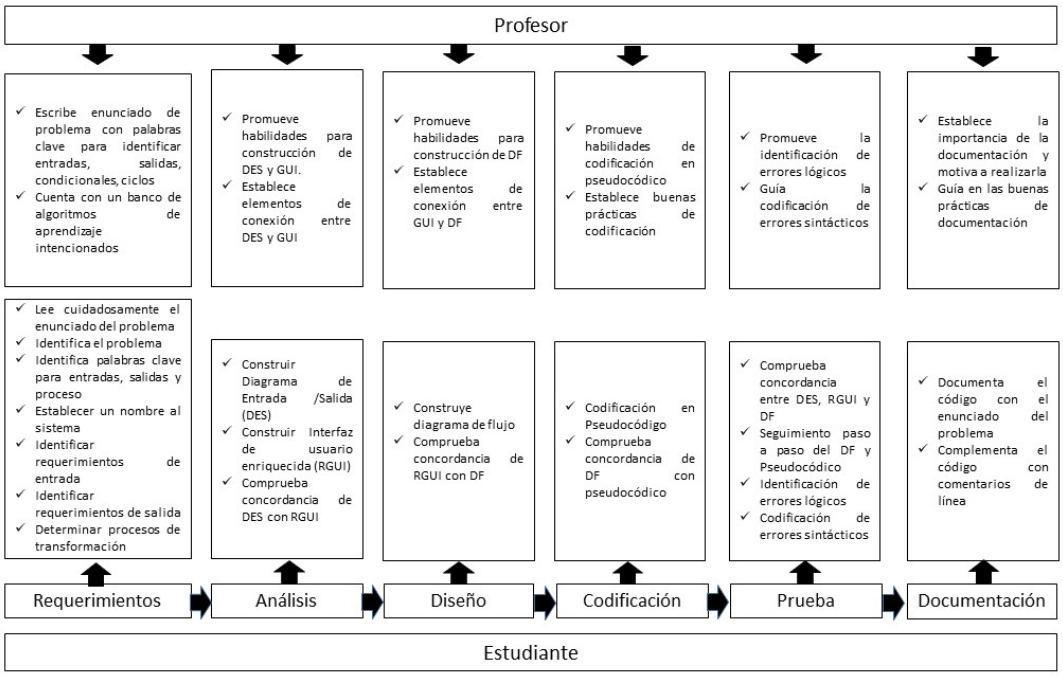

Fig.7. Detalle de actividades en proceso de desarrollo de software

En la figura 6, 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 un Label (por ejemplo, un JLabel en Java o un Label en .Net).

Diseño. El profesor le enseñará al estudiante que después de la fase de recolección de requerimientos y análisis, es necesario la construcción de modelos. 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.

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).

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 multilínea e iniciando con la escritura del enunciado del problema en el mismo script de codificación.

En la figura 7 se presenta el detalle de actividad tanto para el docente 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.

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.

El modelo presentado en este estudio puede tener algunas posibles limitaciones, desde los roles de estudiante y profesor, entre ellas, cuando los estudiantes presentan carencias de las habilidades de pensamiento sugeridas en el modelo, lo ideal sería invitarlo a realizar un curso propedéutico que le ayude a equilibrar sus habilidades, para que el proceso de aprendizaje sea más efectivo. Asimismo, es recomendable el profesor que imparte el curso en lo posible no sea novato, ya que su experiencia en manejo de grupo, en orientaciones didácticas, entre otras, son de mucha importancia para el modelo.

Además, con el modelo de desarrollo de pensamiento

algorítmico de la figura 3, el modelo de enseñanza de la figura 4 y el diagrama de detalle de actividades de la figura 7, que son documentados por extensión en esta investigación, se puede afirmar que este estudio también se puede aplicar en cursos B-learning, E-learning o híbridos.

IV. CONCLUSIONES

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.

El desarrollo de pensamiento algorítmico para abordar un CS1 debe tomar como base los componentes estructurales del proceso de desarrollo de software para obtener en el estudiante resultados de aprendizaje evolutivos.

Los hallazgos presentados en la sección 3, permiten establecer para RQ1 que el desconocimiento de las temáticas y la falta de asociación de los conceptos con la propia experiencia son los reportes más citados, seguidos de los problemas 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. Además, 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, lo que responde a RQ2. Asimismo, la importancia del desarrollo de pensamiento computacional en un CS1 es clave y elemento importante para el proceso de formación del estudiante como futuro programador de computadores, respondiendo a RQ3. Por su parte y como elemento principal de este artículo, 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; lo anterior permite abordar RQ4. También, como resultado de este mapeo, se presentan una serie de consideraciones didácticas, estrategias de enseñanza y aprendizaje para el desarrollo de pensamiento algorítmico en un CS1, lo cual responde RQ5.

Además, la forma de afrontar un CS1 en la formación universitaria aún es diversa frente a los conocimientos impartidos, la didáctica, la metodología y demás factores requeridos para garantizar un aprendizaje efectivo de la disciplina.

Asimismo, la complejidad en un CS1 se incrementa al tratar de que un estudiante articule adecuadamente el manejo de al menos un lenguaje de programación con el modelado de situaciones problema y a la vez con la optimización de las mismas.

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 un estudiante debe tener al iniciar un CS1.

Como trabajo futuro se está desarrollando una herramienta de visualización de software que permitirá mejorar los procesos de abstracción en el desarrollo de pensamiento algorítmico para un CS1.

AGRADECIMIENTOS

Especial agradecimiento a la Universidad del Cauca (Colombia) a través de su grupo de investigación IDIS, la Universidad Castilla – La Mancha (España) con el grupo CHICO y a la Universidad CESMAG (Colombia) mediante el grupo Tecnofília y Luca Paccioli.

REFERENCIAS

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] J. Flores and F. García, *LA VIDA ALGORÍTMICA DE LA EDUCACIÓN : HERRAMIENTAS Y SISTEMAS DE INTELIGENCIA ARTIFICIAL PARA EL APRENDIZAJE EN LÍNEA*, McGraw-Hil., vol. 1. 2023.

[7] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic Mapping Studies in Software Engineering,” *12Th Int. Conf. Eval. Assess. Softw. Eng.*, vol. 17, p. 10, 2008, doi: 10.1142/S0218194007003112.

[8] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Inkman, “Systematic literature reviews in software engineering – A systematic literature review,” *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009, doi: https://doi.org/10.1016/j.infsof.2008.09.009.

[9] B. Kitchenham, T. Dyba, and M. Jorgensen, “Evidence-based software engineering,” in *26 th International Conference on Software Engineering*,

2004, pp. 273–281.
[10] B. Kitchenham and S. Charters, ""Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3," 2007.
[11] B. Kitchenham *et al.*, "Systematic literature reviews in software engineering – A tertiary study," *Inf. Softw. Technol.*, vol. 52, no. 8, pp. 792–805, 2010, doi: https://doi.org/10.1016/j.infsof.2010.03.006.
[12] N. Castro-Gutiérrez, J. A. Flores-Cruz, and F. Acosta Magallanes, "Laboratorio Virtual de Electromagnetismo como estrategia didáctica utilizando el enfoque de aprendizaje situado en ingeniería," *Publicaciones*, vol. 53, no. 2, pp. 255–292, 2023, doi: 10.30827/publicaciones.v53i2.26827.
[13] 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.
[14] J. D. Martínez Díaz, V. Ortega Chacón, and F. J. Muñoz Ronda, "El diseño de preguntas clínicas en la práctica basada en la evidencia. Modelos de formulación," *Enfermería Glob.*, vol. 43, pp. 431–438, 2016, doi: https://doi.org/10.6018/eglobal.15.3.239221.
[15] 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.
[16] 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.
[17] 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.
[18] 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.
[19] 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.
[20] 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.
[21] E. Lamagna, "Algorithmic thinking unplugged," *J. Comput. Sci. Coll.*, vol. 30, no. 6, pp. 45–52, 2015.
[22] M. Kurilov, "Once more about the axioms of programming and about teaching him," *Artif. Intell.*, vol. 3, pp. 4–12, 2015.
[23] 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.
[24] 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.
[25] 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.
[26] 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.
[27] 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.
[28] 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.
[29] 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.
[30] 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.
[31] 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.
[32] 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.
[33] 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.

[34] 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.
[35] S. Gökoğlu, “Algorithm Perception in Programming Education: A Metaphor Analysis,” *Cumhur. Int. J. Educ.*, vol. 6, no. 1, pp. 1–14, 2017.
[36] 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.
[37] 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.
[38] 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.
[39] 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.
[40] 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.
[41] J. Zapotecatl, *Introducción al pensamiento computacional: conceptos básicos para todos*. México D.F.: AmexComp, 2018.
[42] M. Christodoulou, E. Szczygieł, Ł. Kłapa, and W. Kolarz, *Algorithmic and Programming*, Krosno. Rzeszowska, Polonia, 2018.
[43] 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.
[44] 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.
[45] RAE, “Algorítmo,” *Real Academia Española*. 2019.
[46] 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.
[47] 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.
[48] 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.
[49] 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.
[50] 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.
[51] 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.
[52] 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.
[53] A. Sarría, A. Gómez, and A. Granda, “Estrategias didácticas en el proceso de enseñanza- aprendizaje de la programación,” *Rev. Univ. y Soc.*, vol. 13, no. 2, pp. 549–556, 2021.
[54] 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.
[55] 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_influencing_student_learning_of_programming.
[56] 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.
[57] S. Casas and V. Vanoli, “Programación y Algoritmos: Análisis y Evaluación de Cursos Introductorios,” 2007.
[58] 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.
[59] L. P. Baldwin and J. Kuljis, “Learning programming using program visualization techniques.,” 2001, [Online]. Available: http://www.computer.org/portal/.
[60] J. C. Forden, “Sistemas de tareas docentes para la

enseñanza y aprendizaje de la Programación Orientada a Objetos," *Rev. Univ. y Soc.*, vol. 14, no. 1, pp. 480–491, 2022.
[61] 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.
[62] 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.
[63] I. Tamouli, E. Doyle, and M. Huggard, "Establishing structured support for programming students," 2004.
[64] W. Hartman, J. Nievergelt, and R. Reichert, "Kara, finite state machines, and the case for programming as part of general education.," 2001.
[65] 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.
[66] NSF, "Science and engineering indicators 2012," *National Sciience Foundation*, Arlington, VA, USA., 2012.
[67] 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.
[68] W. Dann, S. Copper, and R. Pausch, *Learning to program with Alice. Upper Saddle River, NJ.* 2006.
[69] 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.
[70] A. Smith and C. Johnson, "Understanding Abstraction Challenges in Novice Learners," *J. Educ. Psychol.*, vol. 118, no. 3, pp. 1–21, 2022.
[71] D. Zambrano Salinas, A. Intriago Panchano, K. Camacho Sellán, and E. Ulloa Acosta, "Las emociones que más predominaron en el primer año de pandemia por Covid-19 para estudiantes universitarios novatos," *Edetania. Estud. y propuestas socioeducativos.*, vol. 62, pp. 61–89, 2022, doi: 10.46583/edetania_2022.62.1055.
[72] 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.
[73] G. Bett *et al.*, "Desarrollo de Juegos como Estrategia Didáctica en la Enseñanza de la Programación," *Universidad Tecnológica Nacional, Facultad Regional Córdoba*, 2013. http://conaiisi.frc.utn.edu.ar/PDFsParaPublicar/1/schedConfs/4/120-429-1-DR.pdf (accessed May 22, 2014).
[74] D. Knuth, *Algoritmos fundamentales. Volumen 1*. España: Reverte, s.a., 1980.
[75] 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.
[76] O. I. Buriticá Trejos, "Modelo de enseñanza con aprendizaje colaborativo en estudiantes de programación de computadores," *Vector*, vol. 9, pp. 48–58, 2014.
[77] 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.
[78] 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.
[79] M. Garcia, "Inappropriate Use of Teaching Resources: Implications for Classroom Learning," *J. Educ. Res.*, vol. 127, no. 2, pp. 1–23, 2023.
[80] H. Ramadhan, "Programming by discovery," *J. Comput. Assist. Learn.*, vol. 16, pp. 83–93, 2000.
[81] J. Álvarez, "Pensamiento computacional y multimedia: un cambio en el paradigma educativo," 2013.
[82] P. Thagard, *La mente: introducción a las ciencias cognitivas*. Buenos Aires, Argentina: Katz Editores, 2008.
[83] G. Futschek, *Algorithmic Thinking: The Key for Understanding Computer Science*, vol. 4226, no. November. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.
[84] G. Brassard and P. Bratley, *Fundamentos de Algoritmia*. Madrid, España: Prentice Hall, 1998.
[85] M. Garcia and J. López, "Considerations on Algorithms," *J. Comput. Technol.*, vol. 8, no. 3, pp. 2–27, 2022.
[86] Google, "Exploring Computational Thinking," *Google for education*, 2020. https://edu.google.com/resources/programs/exploring-computational-thinking/# (accessed Jul. 25, 2020).
[87] 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.
[88] V. Michell and T. García, "La importancia del pensamiento computacional en la educación superior / A importância do pensamento computacional no ensino superior," *Brazilian J. Dev.*, vol. 8, no. 6, pp. 48418–48435, 2022, doi: 10.34117/bjdv8n6-377.
[89] 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.
[90] 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.
[91] O. Vorobey, "Teaching Fundamentals of Algorithmization in Grade 5," *Komput. v shkoli ta simi*, vol. 2, pp. 7–10, 2014.
[92] 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.
[93] S. Grover, "Computer science is not just for big kids," *Learn. Lead. with Technol.*, vol. 37, no. 3, pp. 27–29, 2009.
[94] 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.
[95] R. Ruíz, *Una Introducción a la programación estructurada en C*, Primera ed. Santa Fe, Argentina: El Cid Editor, 2013.
[96] P. Halperin, "Development of research on the formation of mental actions," *Psychol. Sci. USSR*, vol. 1, pp. 441–469, 1959.
[97] N. Talyzina, *Educational psychology*. Moscow: Center "Academy," 2006.
[98] L. Landa, *Algorithms and software training. Some questions of the theory and methodology of programming*. Moscow, Russi: Prosveshchenie, 1965.
[99] 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.
[100] 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.
[101] M. Resnick *et al.*, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, p. 60, Nov. 2009, doi: 10.1145/1592761.1592779.
[102] D. Torres, C. Jessup, and N. Margie, "Problemas reales: una alternativa para el desarrollo del pensamiento," *Rev. Arbitr. DEL Cent. Investig. Y Estud. GERENCIALES (BARQUISIMETO - Venez.*, vol. 41, pp. 41–54, 2023.
[103] M. Zhaldak, "Informatics is a fundamental scientific discipline," *Comput. Sch. Fam.*, vol. 2, pp. 39–43, 2010.
[104] I. Minty, "Forms of training organization for the formation of competencies in programming," *Pedagog. High. Middle Sch.*, vol. 41, pp. 91–96, 2014.
[105] 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.
[106] R. Paucar, "Influencia del pensamiento computacional en los procesos de resolución de problemas en los estudiantes de ingeniería de reciente ingreso a la universidad," 2023.