TecnoLógicas
ISSN-p 0123-7799
ISSN-e 2256-5337
Vol. 22, edición especial
Noviembre de 2019
pp. 83-117

Artículo de Revisión/Review Article

Consideraciones en los procesos de enseñanza-aprendizaje para un primer curso de programación de computadores: una revisión sistemática de la literatura

Considerations for the Teaching-Learning Processes of Introductory Programming Courses: A Systematic Literature Review

Javier A. Jiménez-Toledo[1], Cesar Collazos[2] y Oscar Revelo-Sánchez[3]

Recibido: 25 de septiembre de 2019
Aceptado: 19 de noviembre de 2019

Cómo citar / How to cite
J. A. Jiménez-Toledo, C. Collazos, y O. Revelo-Sánchez, "Consideraciones en los procesos de enseñanza-aprendizaje para un primer curso de programación de computadores: una revisión sistemática de la literatura", *TecnoLógicas*, vol. 22, pp. 83-117, 2019. https://doi.org/10.22430/22565337.1520

[1] MSc. en Computación, Facultad de Ingeniería, Universidad Cesmag, Pasto-Colombia. jajime-nez@unicesmag.edu.co
[2] Ph.D. en Ciencias, Facultad de Ingeniería Electrónica y Telecomunicaciones, Departamento de Sistemas, Universidad del Cauca, Popayán-Colombia. ccolla-zo@unicauca.edu.co
[3] MSc. en Computación, Facultad de Ingeniería, Departamento de Sistemas, Universidad de Nariño, Pasto-Colombia. orevelo@udenar.edu.co

Resumen

Los procesos de enseñanza-aprendizaje en la formación de futuros constructores de software han tomado especial importancia en los últimos años, al punto de que esta ya no es una preocupación exclusiva del campo ingenieril, sino en ella convergen profesionales de diversas áreas como la psicología, la pedagogía, el diseño gráfico, las licenciaturas, etc. Aunque existen algunos trabajos desarrollados, no se encuentra un estudio sistemático que recopile las principales problemáticas y las prácticas actuales para afrontar un primer curso de programación. Es por eso que en este artículo se presenta una revisión sistemática de la literatura, con el propósito de exponer los estudios adelantados en los primeros cursos de programación de computadores, en carreas profesionales que forman constructores de software. Con este objetivo, se hizo una exploración en cuatro bases de datos de referencias bibliográficas de publicaciones científicas, en las cuales se hallaron 106 estudios divulgados en los últimos siete años. Tras una detallada inspección, se determinó que 46 de estos coinciden con los criterios de la revisión, lo que da 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, 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. Este artículo es una fuente importante de partida para trabajos futuros que pretendan seguir aportando experiencias que beneficien tanto a estudiantes como a docentes en este complejo campo de la enseñanza-aprendizaje de la programación de computadores.

Palabras clave

Programación de computadores, enseñanza-aprendizaje, revisión sistémica, primer curso de programación.

Abstract

In recent years, software development teaching-learning processes have taken on special importance because they are no longer exclusive of engineering fields; they are also applicable to various areas such as psychology, pedagogy, graphic design, and education. Although the literature includes some studies in this regard, none of them compiles the main problems and current practices of introductory programming courses. Therefore, this article presents a systematic review of literature on introductory programming courses in undergraduate programs. For that purpose, we searched four databases of bibliographical references of scientific publications and found 106 studies published in the last seven years. A detailed inspection determined that 46 of them met the criteria of this review. As a result, we compiled 33 software tools, 36 work strategies, and 18 methodological considerations, as well as important recommendations, experiences, practices, problems, and future trends related to introductory computer programming courses. This article is a starting point for future work that further benefits students and professors in the complex field of computer programming teaching-learning.

Keywords

Computer programming, teaching-learning, systematic review, first programming course.

1. INTRODUCCIÓN

Son muchos los beneficios de aprender a programar computadores, uno de estos es que permite el desarrollo de diversas competencias como el pensamiento crítico, el análisis de conceptos y la resolución de problemas; además, los estudiantes aprenden a trabajar en grupos y a colaborar entre ellos, en su esfuerzo por desarrollar programas ejecutables, mientras se ejercitan en el intercambio de conocimientos y la comunicación de ideas [1]. En razón a esto, 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 en futuros trabajos [2].

En los últimos quince años ha existido una fuerte corriente que propone un mayor rigor en la investigación en el área de la enseñanza de la informática [3].

Este esfuerzo se ha con-centrado mayoritariamente en el aprendizaje de la programación, dado su papel introductorio y central en esta disciplina. Esta corriente investigadora también se aprecia en otras inge-nierías, representadas por la American Society for Engineering Education (ASEE) y el Insti-tute of Electrical and Electronic Engineers (IEEE); sin embargo, ha sido el Special Interest Group in Computer Education (SIGCSE) de ACM el que ha dado el mayor impulso a la ense-ñanza de la informática como campo de investigación (a veces llamado Computing Education Research, CER) [4].

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, cuyos in-convenientes se originan desde el curso de programación inicial, posiblemente desarrollado en su primer año de ingreso a la universidad o institución de formación [5].

Dichos problemas se deben a múltiples circunstancias que surgen por el desconocimiento de conceptos fundamentales de programación, la falta de habilidades para modelar y construir un programa computacional e incluso la poca disciplina al momento de la construcción cognitiva requerida para enfrentar los fundamentos de programación [6].

De lo anterior se puede deducir que la programación de computadores es sin duda un campo que reviste especial importancia en muchos sentidos. La dimensión que abarca el valor de los datos en este momento da cuenta de la magnitud de la programación. Basta con mencionar el alcance del tratamiento de datos automatizados en entidades gubernamentales, bancarias, de salud, empresariales, del sector educativo, entre otras, cuyos sistemas de in-formación o aplicaciones computacionales son de vital importancia para competir en el ejercicio misional propuesto o simplemente para reportar información a sus organismos de vigilancia y control.

Dado el panorama descrito, la responsabilidad recae finalmente en el programador de computadores, por lo cual su proceso de formación debe ser adecuado y riguroso. Esto se convierte en un reto no solo para el docente a cargo, sino para los centros educativos, la industria y la misma la comunidad académica y científica, que deben propender por la búsqueda constante de metodologías, métodos, enfoques, mecanismos y nuevos descubrimientos en los procesos de enseñanza y aprendizaje.

El objetivo principal debe ser motivar y formar al futuro profesional de la industria del software, con miras al bienestar del ser humano y al desarrollo ingenieril y tecnológico.

Este artículo tiene como fin presentar al lector los resultados de una revisión sistémica de literatura acerca de los procesos de enseñanza-aprendizaje, para un

primer curso de pro-gramación de computadores, reportados en la literatura científica. En dicha revisión se en-contraron hallazgos que permiten determinar las problemáticas que enfrentan tanto estudian-tes como profesores y las prácticas actuales propuestas en la formación de programadores de computadores.

El documento cuenta con las siguientes secciones: en la sección 1, se plantea una preám-bulo acerca de la importancia de la programación de computadores y de su proceso de ense-ñanza-aprendizaje; en la sección 2, se hace un acercamiento a la programación de compu-tadores; en la sección 3, se formula y detalla el método de investigación; en la sección 4, como resultado del proceso, se exponen los hallazgos encontrados; en la sección 5, se pone a consideración la discusión de resultados; finalmente, la sección 6 contiene las conclusiones, seguidas de las referencias que soportan el estudio.

2. LA PROGRAMACIÓN DE COMPUTADORES

La programación no es únicamente escribir código fuente, ya que este es el resultado de una serie de actividades previas que le garantizan cualidades como flexibilidad, robustez y concordancia con los objetivos planteados. Por esta causa, un programa informático es una colección de instrucciones, que al ejecutarse efectúa actividades específicas, a través de un sistema de cómputo. Para su escritura y ejecución, necesita 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 [7].

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 garantizan su efectivo aprendizaje. En consecuencia, es necesario incorporar este nuevo modelo mental [8], que exige una constante actualización de acuerdo con las tendencias experimentadas en los recientes enfoques emergentes en este campo.

Por otro lado, la programación de computadores permite resolver problemas puntuales que demandan la mediación de objetos tecnológicos [9] con capacidades de procesamiento autónomo, almacenamiento interno o externo, que permitan brindar respuestas a través de los medios demandados. Todo esto se desarrolla mediante metodologías y modelos específicos validados por toda una comunidad.

La resolución de problemas mediante programación precisa 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 la computación con sus algoritmos, herramientas, técnicas y metodologías de programación [10].

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 hacer tareas específicas, mediante un sistema de cómputo que puede desempeñar hasta acciones dotadas con patrones inteligentes, a través del hardware provisto [11].

3. MÉTODO

Para la revisión sistemática de la literatura presentada en este artículo, se utilizó un enfoque de procesos investigativos de la Ingeniería de Software [12]- [16]. El propósito principal era ofrecer una visión general del área de investigación e identificar la cantidad, el tipo de investigaciones y los resultados disponibles,

para así construir un esquema de clasificación y una estructura válida.

Este enfoque es uno de los más utilizados por los expertos, debido a que genera conocimiento mediante publicaciones contenidas en documentos producto de procesos investigativos [17].

En la Fig. 1, se presentan las etapas del método definido para esta revisión.

3.1 Preguntas de investigación

El objetivo de este estudio fue presentar los resultados de la aplicación de una revisión sistemática de la literatura científica relacionada con la enseñanza/aprendizaje de los fundamentos de programación. Para la construcción del estado del arte, se formularon las siguientes preguntas:

—RQ1: ¿Qué problemas de aprendizaje se han reportado en los estudiantes de fundamentos de programación?

—RQ2: ¿Qué inconvenientes se reportan en la enseñanza de los fundamentos de programación?

—RQ3: ¿Qué herramientas tecnológicas de enseñanza-aprendizaje se utilizan en un primer curso de programación de computadores?

—RQ4: ¿Qué estrategias de enseñanza-aprendizaje se aplican en un primer curso de programación de computadores?

—RQ5: ¿Existen algunas consideraciones metodológicas para enfrentar un primer curso de programación de computadores?

—RQ6: ¿Qué tendencias tiene la programación de computadores que puedan cambiar los actuales procesos de enseñanza/aprendizaje?

La estructuración de estas preguntas se hizo mediante el Modelo PICO con su variante PIPOH [18], cuyos conceptos se detallan en la Tabla 1.

Fig. 1. Estructura de la revisión sistemática de la literatura. Fuente: elaboración propia.

Tabla 1. Definición de los conceptos generales con PIPOH. Fuente: elaboración propia.

Criterio	Descripción
Población	Enseñanza/aprendizaje de los fundamentos de programación
Intervención	Problemas de enseñanza/aprendizaje, procesos de enseñanza/aprendizaje, herramientas
Resultados	Publicaciones de artículos o libros, en los cuales se citen estudios de enseñanza/aprendizaje de los fundamentos de programación
Profesionales	Ciencias de la Computación
Contexto	Académico

3.2 Búsqueda

La estrategia de búsqueda propuesta consistió en la exploración de los términos generales en bases de datos especializadas.

En las bases de datos seleccionadas, se aplicaron los términos de búsqueda combinados con sinónimos, con el propósito de cubrir una mayor cantidad de documentos a evaluar. En la Tabla 2, se exponen los términos con sus respectivos sinónimos y filtros.

El proceso de búsqueda se adelantó en las bases de datos mencionadas en la Tabla 3, con base en campos como: título, palabras, resumen y documento completo.

En la Tabla 4, se muestran las cadenas de búsqueda resultantes para cada término, junto con sus sinónimos y sus correspondientes filtros.

Cabe anotar que la búsqueda planteada se llevó a cabo desde el año 2012, debido a la calidad de los aportes académicos y científicos reportados en los bancos de datos validados por esta misma comunidad, que actualmente son considerados como elementos primordiales en el campo de la enseñanza/aprendizaje de los fundamentos de programación.

Tabla 2. Término, sinónimos y filtros para componer la cadena de búsqueda
Fuente: elaboración propia.

Término	Sinónimos	Filtros adicionales
Teaching-learning of computer programming	Teaching in computer programming	Education
	Learning in computer programming	Computer's science
	Programming teaching environments	
	A first computer programming course	Software
	Teaching-learning strategies in computer programming	Year of publication> = 2012

Tabla 3. Término principal y base de datos. Fuente: elaboración propia.

Término	Base de datos
Teaching-learning of computer programming	Redalyc
	IEEE Xplorer
	Springer
	ISI (Web of Science)

Tabla 4. Cadena de búsqueda. Fuente: elaboración propia.

Término principal	Cadena de búsqueda
Teaching-learning of computer programming	("Teaching-learning of computer programming" OR "Teaching in computer programming" OR "Learning in computer programming" OR "Programming teaching environments" OR "A first computer programming course" OR "Teaching-learning strategies in computer programming") AND (education Or Software Or Computer's science) AND (publication year >= 2012)

3.3 Selección

Todos los estudios contemplados en esta revisión fueron analizados; en su proceso de evaluación, se tuvieron en cuenta los ítems: título, palabras clave, resumen, introducción, antecedentes, estado del arte, metodología, resultados y conclusiones.

Así mismo, se definieron criterios de inclusión y exclusión en su revisión.

El criterio de inclusión definido es: documento científico relacionado con estudios de procesos de enseñanza aprendizaje de los fundamentos de programación para estudiantes de carreras universitarias o sus sinónimos de búsqueda. De la misma manera, los criterios de exclusión definidos son: estudios que no tengan su correspondiente cita bibliográfica, el documento no contiene los términos o sinónimos de búsqueda, solo experiencias con primeros cursos de programación de computadores y los documentos no están disponibles para descarga.

La selección de las fuentes primarias se hizo en cuatro momentos (debido a que se consideraron cuatro bases de datos especializadas), cada uno de estos con tres fases: fase 1: eliminación de artículos duplicados; fase 2: eliminación de artículos no descargables y fase 3: aplicación de criterios de inclusión y exclusión.

Debido a que en esta revisión sistemática se utilizaron cuatro bases de datos con sus sinónimos y filtros —en cada una de las cuales se aplicaron tres fases—, fue necesario hacer la consulta en diversas fechas como se observa en la Tabla 5.

Como resultado de la búsqueda en las bases se encontró un total de 106 estudios y al aplicar los criterios de inclusión y exclusión se obtuvo un total de 50 documentos.

Tabla 5. Fechas de consulta y descarga de archivos
Fuente: elaboración propia.

Base de datos	Fecha de búsqueda
Redalyc	27/04/2018
IEEE Xplorer	22/05/2018
Springer	14/06/2018
ISI (Web of Science)	04/07/2018

3.4 Evaluación de calidad

En el proceso de evaluación de calidad de los documentos aquí seleccionados, se tuvieron en cuenta siete criterios: procedencia de las fuentes, relevancia del contenido, impacto del estudio, objetivo de la investigación, contexto del estudio, objetividad del diseño metodológico y rigurosidad científica en el análisis de los datos. Estos siete criterios obedecen a tres elementos importantes de la gestión de calidad: planificación, organización y control.

Además, la evaluación realizada sobre los escritos en mención implicó la lectura y análisis completo de los 50 documentos con sus procesos de eliminación de artículos duplicados, eliminación de artículos no descargables y la aplicación de criterios de exclusión e inclusión, cuyo proceso se expone en la Tabla 6.

En la Tabla 7 se presentan las referencias de los 50 artículos estudios contemplados en esta revisión sistemática ordenados por su año de publicación.

3.5 Extracción de datos y síntesis de resultados

Dado que el objetivo principal de este estudio sistemático es determinar el estado del arte de los procesos de enseñanza-aprendizaje de los fundamentos de programación, una vez hecha la búsqueda de los términos en cada base de datos y aplicadas en cada una de ellas las tres fases, a continuación, se presentan los hallazgos en los archivos pertinentes con sus correspondientes citas de autor.

Tabla 6. Evaluación de calidad en procesos de búsqueda y selección. Fuente: elaboración propia.

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	1	8	14	ISI (Web of Science)
Total	106	15	41	50	

Tabla 7. Estudios incluidos en revisión sistemática. Fuente: elaboración propia.

Año	Cantidad	Referencias
2012	11	[5] [9] [22] [28] [34] [48] [73] [80] [86] [104] [113]
2013	5	[36] [40] [42] [43] [122]
2014	11	[1] [28] [45] [46] [47] [51] [98] [105] [114] [115] [123]
2015	10	[19] [30] [41] [67] [75] [78] [92] [109] [111] [112]
2016	6	[18] [21] [26] [74] [89] [90]
2017	2	[4] [91]
2018	5	[17] [52] [53] [83] [107]
Total	50	

4. RESULTADOS

Los resultados obtenidos con el proceso de revisión sistémica de la literatura se presen-tan exponen a continuación, organizados en cinco secciones para afrontar un primer curso de programación de computadores: herramientas utilizadas en la enseñanza-aprendizaje de la programación, estrategias, consideraciones metodológicas, recomendaciones y tendencias de la programación de computadores.

4.1 Problemas en la enseñanza/aprendizaje de la programación de computadores

Tanto la enseñanza como el aprendizaje de la programación de computadores son tema de estudio de varios autores. Debido a las dificultades reportadas, se han adelantado diversas investigaciones que proponen la construcción de metodologías, técnicas y herramientas que tratan de aportar soluciones a los problemas encontrados en los dos ámbitos [19].

El aprendizaje de la programación computacional es complejo para muchos estudiantes noveles y, al mismo tiempo, se convierte en un desafío para los docentes [20]. Para afrontar la asimilación de los fundamentos de programación son necesarias determinadas habilidades cognitivas, entre ellas, la abstracción, las aptitudes lógico-matemáticas y la capacidad para solucionar problemas algorítmicamente. Los factores de motivación y didácticas aplicadas son importantes para el proceso de enseñanza, al abordar los conceptos fundamentales de programación en el aula de clase [21].

Además de generar un nuevo conocimiento, el aprendizaje de la programación se consolida como una herramienta eficaz para la solución de problemas [19].

La literatura científica reporta problemas tanto en el proceso de enseñanza guiada por los docentes como de aprendizaje por parte de los estudiantes, al enfrentar el estudio de los fundamentos de programación. Las principales dificultades

en la enseñanza/aprendizaje se describen a continuación.

4.1.1 Dificultades en el aprendizaje de la programación de computadores

La asimilación de los fundamentos de programación o 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, entre otros [22]-[25].

Así mismo, a pesar de culminar un curso de programación de computadores, algunos estudiantes no adquieren las habilidades básicas [21], de lo que se infiere que la programación es una disciplina difícil de aprender. Informáticos, pedagogos y psicólogos llevan décadas investigando las dificultades que encuentran los alumnos para su aprendizaje [4].

El proceso de aprendizaje requerido en la programación de computadores se considera una tarea difícil, pues los estudiantes deben tener destrezas en el campo cognitivo de orden superior como la resolución de problemas, el desarrollo y la aplicación de modelos mentales o matemáticos y la generación de algoritmos.

A su vez, deben estar dispuestos a aprender diversas sintaxis y semánticas requeridas para codificar programas de computadores [26], lo que muchas veces deriva en que algunos sientan frustración y, en casos extremos, se retiren de sus programas de estudio [26]-[28].

Dann, Cooper y Pausch [29] determinan la existencia de cuatro elementos que complican el aprendizaje de la programación: metodologías de enseñanza inapropiadas para el tratamiento de la sintaxis en la codificación de programas; la falta de un resultado tanto de los cálculos como del seguimiento de la estructura del programa 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 internacional se han llevado a cabo muchas investigaciones que reportan carencias frente a la asimilación de los diversos conceptos fundamentales de programación, pero son escasos los estudios que expongan de forma clara y contundente el fundamento científico de la estructura de esta destreza [30]. A pesar de que en la actualidad existen varias metodologías, enfoques, métodos y herramientas de enseñanza-aprendizaje de los fundamentos de programación, no hay una solución que satisfaga todas las necesidades por consenso [31], [22] y que logre cumplir con éxito los objetivos de los estudiantes del siglo XXI [1].

Aun cuando la familiarización con las computadoras sugeriría que el aprendizaje podría ser más fácil hoy en día, en la actualidad, los estudiantes continúan enfrentando dificultades en cursos de programación [1]. Los amplios estudios de investigación adelantados en las últimas dos décadas indican que estas dificultades aún están presentes y los estudiantes parecen estar menos interesados en la programación [32], [1].

Baldwin y Kulijis [21], [33] identifican otros factores que obstaculizan el aprendizaje de la programación de computadores en los estudiantes como en el caso de los complejos procesos cognoscitivos requeridos en las primeras etapas de estudio, ya que estas involucran tareas de orden superior del pensamiento como la planificación, el razonamiento y la misma solución de problemas. A esto obedece la importancia de desarrollar ciertas habilidades del pensamiento antes de enfrentar un curso de programación de computadores; esto se convierte en un factor decisivo para el correcto aprendizaje de la lógica computacional [21].

En este momento, se cuenta con muchos recursos didácticos disponibles

para el aprendizaje de la lógica de programación y su codificación; no obstante, aún no existen entornos completos ni tampoco metodologías precisas que conlleven un aprendizaje efectivo con base en las características individuales de los estudiantes [34].

Otra dificultad de los estudiantes frente a un primer curso de programación es el manejo de una terminología totalmente desconocida en su entorno experiencial; por ejemplo, la concepción de una variable, el manejo de memoria asignado a un estado o, simplemente, el concepto de tipado de datos, hacen aún más confuso el aprendizaje en esta primera etapa [35].

En el mismo sentido, algunos autores argumentan que la complejidad presentada en la estructura sintáctica del código también dificulta los procesos de aprendizaje; de igual forma ocurre con la mala calidad de los instrumentos de aprendizaje existentes y el débil desarrollo de las destrezas necesarias para afrontar un problema [21].

Asimismo, los estudiantes encuentran inconvenientes en la interpretación de los enunciados planteados, que pueden ser consecuencia de problemas relacionados con la deficiencia en procesos de abstracción [36]. En razón a esto, algunos estudiantes de los primeros cursos que no logran alcanzar las habilidades necesarias para la comprensión de los fundamentos de programación, adquieren una actitud de rechazo ante el proceso de aprendizaje [21], [37]-[39].

A parte de los problemas mencionados, tanto universidades como instituciones educativas aún siguen discutiendo sobre cuál debe ser el paradigma de programación apropiado para un primer curso de programación de computadores.

Esta discusión se centra principalmente en aquellos de mayor utilización como el paradigma estructurado, el orientada a objetos, el orientado a eventos o el funcional [40]. El proceso de aprendizaje en cada uno de estos, 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 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 desarrollo, 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 [41].

Evidentemente, en un periodo de tiempo corto, el estudiante se enfrenta a una cantidad de nuevos y extraños conceptos que terminan haciendo más compleja la adopción de los estamentos necesarios para la construcción de programas computacionales [42].

Otra de las dificultades para el aprendizaje de la POO son las constantes actualizaciones de los entornos de programación, debido a su alto contenido profesional, que abarca una amplia disponibilidad de herramientas que resultan desconcertantes para un estudiante novato [41]. Igualmente, los métodos de estudio inadecuados se convierten en un gran problema al momento de consolidar la lógica del pensamiento y el desarrollo de habilidades cognitivas requeridas en la codificación de un programa de computador [43].

De acuerdo a la revisión sistemática de la literatura en estudio, en la Tabla 8 se exponen los principales hallazgos en torno a las dificultades presentadas por los estudiantes en el aprendizaje de la programación de computadores y el análisis de sus principales causas y efectos.

Tabla 8. Dificultades, causas y efectos del aprendizaje de la programación en un primer curso
Fuente: elaboración propia.

Dificultades	Posibles causas	Principal efecto
Motivación por el aprendizaje [22]-[25]	Complejidad en las temáticas	Baja asimilación de conceptos Bajo rendimiento académico
Métodos de estudio inapropiados [29][43]	Desconocimiento de los métodos de estudio	Aprendizaje memorístico que no aporta al desarrollo del pensamiento lógico
Pocos niveles de abstracción [29][36]	Falta de experiencia o preparación de su modelo mental	No relaciona los conceptos con su entorno experiencial
Escasas habilidades de pensamiento computacional [21][29]	Bajo desarrollo del pensamiento lógico No existencia de procesos de selección acordes con perfiles de aspirantes	Limitación para desarrollar procesos de programación
Debilidad para la resolución de problemas[21][26][33]	Débil formación al respecto	Poca capacidad de generación de alternativas para resolver problemas
Bajos niveles de desarrollo de pensamiento matemático[26]	Problemas de comprensión matemática Bajos niveles académicos en el componente matemático	Dificultad en la solución de problemas
Débil proceso de conocimiento procedimental [26]	Falta de preparación en los niveles educativos anteriores	Dificultad para construir algoritmos
Comprensión de la metodología de enseñanza del docente [29]	Problemas didácticos en la metodología de enseñanza del docente	Débil asimilación de los conceptos enseñados
Comprobación de resultados de manera instantánea [29]	Construcción de algoritmos, procedimientos y demás en niveles solo de representación como por ejemplo el papel	Baja motivación, confusión
Cantidad de conceptos nuevos para asimilar [35][41]	Curos de corta duración para un primer nivel de programación	Confusión en la implementación de conceptos
Estudio en recursos de aprendizaje inapropiados [21][34]	Recursos didácticos que no contemplan los diversos estilos de aprendizaje	Limitación en el autoaprendizaje
No comprensión de enunciados a resolver [36]	Bajos niveles de comprensión de lectura Inapropiada redacción de enunciados por parte del docente	La solución presentada (si la logra diseñar) dista de la necesidad planteada
Inadecuada articulación de temáticas [40][41]	No existe un verdadero consenso de la forma de abordar un primer curso de programación	Choque de estructuras mentales del estudiante conforme avanzan los diversos cursos de programación

4.1.2 Inconvenientes en la enseñanza de la programación de computadores

La literatura científica reporta varios inconvenientes respecto a la enseñanza de la programación de computadores. El primero de estos es la incertidumbre, al abordar un primer curso, frente a la secuencia de los saberes como *object-first* o *procedural-first*; aunque existen algunos estudios, aún se hace complejo tomar una decisión soportada totalmente con elementos de juicio que determinen la mejor alternativa [44].

A su vez, la enseñanza de la programación de computadoras establece un reto para el docente, quien, aparte del proceso metodológico, debe poner a prueba

la acción didáctica a fin de contar con recursos significativos que guíen al estudiante de manera adecuada en la asimilación de los conceptos necesarios.

Esta es una tarea compleja si se tiene en cuenta que los estudiantes son novatos y muchos no poseen experiencia alguna en programación [40].

Infortunadamente, los mecanismos de instrucción en la iniciación a la programación de computadores tienen un proceso limitado de acuerdo con los avances científicos y tecnológicos vivenciados. 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 vieja escuela, basados en modelos de imitación y caracterizados por la exposición de una solución propuesta por el docente ante un determinado problema, con la esperanza de que el estudiante lo implemente en su modelo mental [45].

De acuerdo al reporte científico publicado en la literatura, la enseñanza de la programación es el área en la que se concentra la mayor preocupación [46].

Los escritos existentes afirman que mayormente son los profesores quienes no inducen procedimientos adecuados que conduzcan a la búsqueda clara de una solución a los problemas de 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. De igual forma, no proponen mecanismos introspectivos de evaluación y, finalmente, el aprendizaje se reduce a la simple explicación brindada en ese momento por el profesor, quien acude al planteamiento de situaciones demostrativas que no dan cuenta de un proceso cognitivo completo, requerido para que el estudiante asimile plenamente el conocimiento [47].

Entre las metodologías existentes para enseñar a programar, hay una que consiste en hacer que el estudiante resuelva una cantidad importante de ejemplos y situaciones problemáticas, con el propósito de hacer codificación, probar sus diseños y corregirlos, hasta que el ejercicio quede completamente correcto. Muchas veces lo anterior genera que el estudiante dedique mucho tiempo a resolver problemas sintácticos del lenguaje y, en ocasiones, a establecer una disputa absurda con el computador, lo que deriva en acciones preocupadas por sobrepasar dicho error, en lugar de determinar un procedimiento lógico enfocado en la solución del problema planteado y no en el manejo del lenguaje [48].

Por otro lado, desde hace mucho tiempo, la enseñanza de los fundamentos de programación de computadoras se ha convertido en un desafío tanto para los mismos docentes como para los profesionales de diversas áreas de las ciencias sociales.

Esto ocurre en tanto que fortalece el dominio y conocimiento de uno o más lenguajes de programación y permite desarrollar destrezas para resolver problemas, construir algoritmos que modelen las soluciones planteadas y determinar la validez de dichas soluciones, con aquellos estudiantes de un curso introductorio de programación que, en muchos casos, ingresan por primera vez a un centro de formación profesional o tecnológico [40].

Como resultado de la revisión sistemática de la literatura, en la Tabla 9 se presentan los hallazgos sobre los inconvenientes que enfrentan los docentes en los procesos de enseñanza de la programación de computadores y el análisis de sus principales causas y efectos.

Tabla 9. Inconvenientes, causas y efectos de la enseñanza de la programación en un primer curso
Fuente: elaboración propia.

Inconvenientes	Posibles causas	Principal efecto
Falta de conceso para un primer curso [44]	Inexpertica del docente Actualización de currículos	Desaprovechar el desarrollo de habilidades fundamentales en un primer curso
No recomendación de métodos de estudio[47]	Docentes 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 [45][47][48]	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[40][46][47]	Recursos didácticos que no contemplan los diversos estilos de aprendizaje	Limitación en el proceso de aprendizaje por parte del estudiante

4.2 Herramientas utilizadas en la enseñanza-aprendizaje de la programación de computadores.

La revisión sistémica permitió determinar las herramientas de software utilizadas para la enseñanza-aprendizaje en un primer curso de programación de computadores clasificadas en cinco categorías como lo muestra la Tabla 10.

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 las siguientes herramientas:

FLowchart INTerpreter o FLINT [49]: Flowchart Interpreter o también FLINT, permite la representación de algoritmos mediante diagramas de flujo con sintaxis minimalista. FLINT facilita la retroalimentación continua de su código interpretado, y cuenta con herramientas de seguimiento propicias para programadores iniciales [26].

Raptor [50]. 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 [26].

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 [51].

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 [51].

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 con forme avanza la secuencia de instrucciones, permitiéndole al estudiante determinar paso a paso el comportamiento del código fuente [52], [53].

Tabla 10. Categorías y herramientas para un primer curso de programación de computadores
Fuente: elaboración propia.

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

Herramientas de evaluación automática. Estas herramientas desarrollan procesos de verificación de instrucciones de forma estática o en ejecución tanto en editores de código como en editores gráficos. En esta categoría se encuentran:

—TRY [54]: suministra al estudiante retroalimentación de forma instantánea y compara paso a paso lo ejecutado con lo esperado. Además, el estudiante puede hacer varios intentos para alcanzar su respuesta, bajo la restricción a cierto número de ensayos, mediante los cuales se busca invitarlo a analizar sus procedimientos antes de intentar nuevamente [20].

—PSGE [55]: surge como resultado de TRI 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. Aunado a lo anterior, este ciclo cuenta con tres etapas: especificación de tareas, un módulo para distribuir tareas y probar programas acordes a un plan determinado [20].

—TRAKLA [56] permite evaluar automáticamente algoritmos; mediante su interfaz gráfica de usuario, el estudiante construye sus propios diseños, bajo la limitación a un determinado número de intentos de ejecución, establecidos en su configuración inicial con el fin de evitar que abuse de la técnica de prueba y error [20].

—CourseMaker [57]: es auto compilado en Java y posibilita la configuración de seguimiento dinámico de interpretación de código y de pruebas estáticas de compilación. Asimismo, cuenta con un módulo de registro del proceso de compilación en MySQL [20].

—AutoLEP [58]: hace un análisis estático en códigos, cuyas pruebas dinámicas no son suficientes, incluido el manejo de marcadores en puntos específicos que requieren análisis de comportamiento especial [20].

—Visual DaVinci: es un entorno integrado de desarrollo (IDE) de programación para la representación y ejecución de algoritmos, con el objetivo de facilitar la enseñanza y el aprendizaje en los primeros cursos de programación de computadores. Fue desarrollado por la Universidad Nacional de La Plata [19].

Juegos educativos centrados en la enseñanza de una unidad específica de aprendizaje [1]. Este enfoque toma como base los juegos ya construidos para la enseñanza de los fundamentos de

programación. A esta categoría pertenecen las siguientes herramientas de software.

—Catacumbas: es un juego tridimensional multijugador, que tiene como objetivo enseña a los estudiantes cómo declarar variables y usar declaraciones y bucles *if* simples y anidados [1]. El juego registra puntajes de experiencia para cada alumno y proporciona mensajes explicativos como un mecanismo de andamiaje [59], [60].

—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 [1]. Los estudiantes deben completar una serie de misiones para progresar en la trama del juego; con este objetivo, finalizan las líneas de código que resultarán en un programa ejecutable. De esta forma, aprenden el algoritmo de ordenación rápida junto con bucles simples y anidados con el uso de un microlenguaje [59], [60].

—EleMental: la recurrencia: es un juego tridimensional que tiene como objetivo enseñar a los estudiantes a ejecutar la recursión y la búsqueda transversal en profundidad, para lo cual deben utilizar el lenguaje de programación C # [1]. Dos avatares, llamados Ele y Cera, ayudan a los estudiantes durante el juego de varias maneras; por ejemplo, una vez que se escribe el código, 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 [61].

—Castillo de Wu: es un juego de rol bidimensional que tiene como objetivo enseñar a los estudiantes *loops* y arreglos, por medio de actividades interactivas [1].

El juego faculta 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 [62].

—Robozzle: es un juego de acertijos en línea que proporciona una serie de comandos predefinidos, listos para usar y no muestra ningún código real [1].

Los usuarios pueden ejecutar sus funciones y ver cómo se mueve su héroe en todo el mundo; por lo tanto, pueden detectar fácilmente los errores que han cometido y volver a programar en consecuencia [63].

—LightBot: es análogo a Robozzle, corresponde a un juego basado en acertijos y se puede jugar en línea. Incluye un conjunto de comandos, con los cuales el estudiante establece acciones secuenciales, aunque no es un lenguaje de programación [64].

—TALENT: utiliza un micro lenguaje para la enseñanza algorítmica de enunciados y bucles. Por cada jugador, se usa un avatar en forma de arqueólogo, mediante el cual el estudiante interactúa con el entorno virtual, desarrolla actividades específicas y recolecta elementos para luego exponerlos en un museo [65].

—Gidget [66]: es una herramienta de programación elemental que consiste en programar en un robot la corrección de fallas para poder completar misiones establecidas previamente [67].

Juegos educativos centrados en la enseñanza de unidades múltiples de aprendizaje. Al igual que en la anterior categoría, se toman como base algunos juegos ya desarrollados para adelantar el proceso de enseñanza. En esta categoría se suscriben:

—Robocode: es un entorno bidimensional que tiene como objetivo enseñar programación de computadoras usando el lenguaje Java. El juego se compone de un editor de programación, robots y una arena virtual, y los estudiantes deben programar un robot que compita contra otros en la arena [1].

Durante su construcción, el robot hereda métodos básicos que luego pueden ser extendidos por los estudiantes, de acuerdo con el comportamiento que esperan de sus robots en la arena [68].

—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 y la utilización del lenguaje de programación Java. La metáfora del juego consiste en que el estudiante debe construir un robot, el cual lucha en una arena virtual con otros robots, mediante la escritura de líneas de comando en un entorno de desarrollo integrado [69].

—Prog & Play: es un entorno de juego en el que los usuarios programan sus avatares, que son héroes y pueden conformar estrategias entre sí para poder subsistir la mayor cantidad de tiempo.

Permite al estudiante elegir el idioma de codificación en Scratch, C, Ada, OCaml y Compalgo [70].

—PlayLogo3D: es un juego de roles tridimensional, que hace posible la interacción entre múltiples usuarios y tiene como objetivo enseñar conceptos básicos de programación informática estructurada [1]. El mundo virtual consiste en la nave espacial X-15, localizada en una constelación de la galaxia de Andrómeda, donde cada año se celebra un concurso entre pilotos-robots [71].

—Gidget: es un juego basado en la web, en el que los estudiantes pueden programar utilizando un lenguaje de programación simplificado, creado específicamente para el juego, con el fin de aprender a diseñar y analizar algoritmos básicos [1]. Un robot llamado Gidget tiene problemas con una parte de su software y, por lo tanto, no puede completar sus tareas; los estudiantes son llamados para ayudar a Gidget, ya sea para arreglar las líneas de código incorrectas o completar el código que falta dentro de los programas [72].

—Scratch: Brennan y Resnick [73], [74] lo describen como un entorno de programación para la construcción de historietas —aplicaciones interactivas—, que cuenta con una enorme comunidad en línea para compartir con otros usuarios.

—Snap! [74], [75]: es una extensión de Scratch y es un lenguaje de programación visual, que presenta opciones de configuración para abordar cursos iniciales de programación en cualquier nivel educativo.

—Train B&P [76]: mediante un sistema basado en ferrocarriles, apoya los procesos de aprendizaje de los fundamentos de programación [67].

—Entorno Cubik [77]: posee un editor y traductor de algoritmos a código fuente, que faculta la escritura de programas con enfoque estructurado o modular, ya que se fundamenta en el paradigma imperativo [78].

Ambientes colaborativos. Son entornos de aprendizaje diseñados para la adecuada interacción entre estudiantes, con el objeto de generar procesos de aprendizaje comunes. Entre los ambientes colaborativos reportados para un primer curso de programación están:

—EclipseGavab [79]: es una versión de Eclipse, personalizada y pensada en el ejercicio docente, que combina el aprendizaje colaborativo con el enfoque de aprendizaje basado en proyectos; además, el estudiante puede escribir su código en lenguajes Pascal, C y Java [80].

—Virtual Programming Lab (VPL) [81]: es un laboratorio de programación soportado en Moodle, lo cual flexibiliza el desarrollo de software de manera virtual al aprovechar 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 [80].

—Ambiente Instruccional SABATO [82]: es una herramienta que combina el aprendizaje colaborativo apoyado por computador, conocido por sus siglas CSCL (Computer Support for Collaborative Learning), y el Aprendizaje Basado en Problemas, bajo las siglas ABP [80].

—COLLECE: diseñado 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 código, control y asignación de tareas, entre otras [4].

—COLLECE 2.0: es una extensión de COLLECE, que integra el IDE de Eclipse en un entorno con varios *plugins* como recursos para programadores que mejoran el contexto colaborativo de su antecesor [4] [83].

Como resultado del proceso de la revisión literaria, se presenta en la Tabla 11 un análisis más amplio de las herramientas utilizadas en un primer curso de programación de computadores.

4.3 Estrategias utilizadas en la enseñanza-aprendizaje de la programación de computadores.

La revisión sistemática reportó las siguientes estrategias, tipificadas según su aplicabilidad en tres tipos, propuestas por los autores del presente estudio como lo muestra la Tabla 12.

Sistemas de visualización. La visualización en procesos de software ayuda a representar de manera gráfica el contenido de un algoritmo o código fuente [84]. Hoy en día hay una amplia variedad de sistemas de visualización de algoritmos y programas que apoyan los procesos tanto de enseñanza como del fundamento de programación [53]. Los sistemas de visualización permiten además simbolizar ideas abstractas mediante elementos gráficos [85], para guiar al estudiante de una forma más simple tanto para la evaluación de su modelo como para la cimentación de su constructo en búsqueda del desarrollo de pensamiento algorítmico.

Marcos de trabajo. Frittelli et al. [40] 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 un primer curso de programación.

Estrategias de intervención. Silva et al proponen combinar técnicas de aprendizaje colaborativo con programación de pares [26] para un primer curso de programación de computadores 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 [86], [17].

Pair programming es una propuesta pedagógica que se ha estudiado de forma amplia [87], [88] 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 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 [89], [17].

Entorno virtual colaborativo inteligente. Es la integración de los Entornos Virtuales Colaborativos con la Inteligencia Artificial [17]. 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 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 [90].

Tabla 11. Características de las herramientas computacionales para un primer curso de programación
Fuente: elaboración propia.

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

Tabla 12. Tipificación de estrategias para un primer curso de programación
Fuente: elaboración propia.

Estrategias	Colaborativas	Centradas en el aprendizaje	Centradas en la enseñanza	Pueden 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
Lenguaje de programación actual			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

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) [17]. 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 docente y el estudiante [91].

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 [17].

La construcción de conocimiento se fundamenta en la discusión social generada como base del razonamiento pedagógico incorporado [92].

Paradigma de programación y lenguaje de programación. Robins et al [93] y Pears [94] resaltan la importancia de enseñar un primer curso de programación combinando un paradigma con un lenguaje de programación de computadores.

Programación de procedimientos. Establece un estilo de aprendizaje sencillo ideal para un novel programador [95]. Lenguaje de programación actual. De Raadt et al. [96] junto con Dingle y Zander [97] proponen el iniciar con un lenguaje de programación acorde a la actualidad de los tiempos y a las necesidades de la industria.

Lenguajes con una sintaxis simple. Koulouri et al [98] con Mannila y De Raadt [99] recomiendan el iniciar con un lenguaje de programación Python o Eiffel los cuales tienen sintaxis simples ya que facilitan el aprendizaje en las condiciones de un primer curso de programación.

Curso de pre-programación. Silva et al [26], Davies et al. [100] y Rizvi et al [101], 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 [26] recomienda el uso de herramientas de soporte de programación con estudiantes noveles para así mejorara los procesos de compresión de las temáticas estudiadas.

Juegos serios. Un número creciente de docentes / investigadores proponen la incorporación de juegos educativos (o juegos serios) en la enseñanza de la programación de computadoras con el objetivo de reforzar la motivación instintiva a través de estudiantes desafiantes, despertando su curiosidad y dándoles una sensación de control e imaginación [1].

Juegos educativos. Una propuesta interesante para aliviar los problemas enfrentados es la incorporación de juegos educativos dentro de cursos de programación informática [1].

Pseudolenguajes. Su utilización es importante en los primeros cursos de programación, estos tienen como ventaja que se puede configurar en idiomas locales y algunos se complementan con editores gráficos y pueden hasta generar código fuente en lenguajes de programación formal [67].

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 [67].

Entornos. Permite la construcción de entornos simulados desde cero mediante procesos algorítmicos que incluyen modelamiento matemático que posteriormente el estudiante manipula y edita de acurdo a los objetivos planteados [67].

Creación de videojuegos. Consiste en creación de un entorno de juego partiendo desde cero, contemplando adecuadamente metodologías que permitan integrar las buenas prácticas de programación ya conocidas [67]. La construcción de entornos de juego básicos permite una motivación mayor en el estudiante al momento de enfrentar un primer curso de programación [40]. Desarrollo de Modelos mentales.

Baldwin y Kulijas [33] proponen que es posible en un primer curso de programación de computadoras 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 de la espiral Herbert [102] establece que la enseñanza más adecuada de los fundamentos de programación a 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 [21].

Máquinas de estados finitos. Para la enseñanza de los fundamentos de programación Hartman, Nievergelt y Reichert [39], 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 de los fundamentos de programación [21].

Analogías. Es una importante técnica utilizada por muchos docentes para la enseñanza de los fundamentos de programación, 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 [103]. Dunican [35] propone analogías para enseñar declaraciones a través de juguetes infantiles; para el maneo de listas con cajas, y el concepto de matriz mediante un casillero de correspondencia [21].

Ambientes colaborativos. Los entornos colaborativos logran potenciar tanto el auto aprendizaje como fortalecer el razonamiento y el aprendizaje colaborativo [104]. Por esto, los ambientes colaborativos permiten que el aprendizaje de los fundamentos de programación se realice en un entorno con ciertos niveles de interactividad entre los estudiantes que beneficia directamente los procesos de adquisición de conocimiento [80].

Metáforas. Son consideradas como una interesante herramienta didáctica y facilitan la enseñanza de un concepto abstracto [105]. En el ámbito universitario ha tomado interés la utilización de metáforas especialmente en los primeros cursos de programación con el propósito de contextualizar al acercar al estudiante a un concepto abstracto [105], [106], 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. [107].

Clases magistrales y de laboratorio [85]. Son las más utilizadas en la mayoría de universidades e instituciones educativas, 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 [108].

Robots. Se han convertido en didácticas del aprendizaje constructivo en los cursos iniciales de programación ya que posibilitan la experimentación en tiempo real [85]. Uno de los mayores representantes en esta categoría son los kits de Robots Lego Mindstorms NXT mediante los cuales algunos docentes enseñan los fundamentos de programación [109].

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 tiene como actividades: la búsqueda de información, seleccionarla, su organización y la

resolución de problemas [110]. 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 [85].

Realidad Aumentada. La inclusión de la realidad aumentada como herramienta didáctica permite motivar de una manera fácil la atención del estudiante y combinándola adecuadamente con la complejidad de los conceptos que debe enfrentar en un primer curso de computación se obtienen interesantes resultados que favorecen el aprendizaje [111].

Realidad Mixta. Facilita el aprendizaje activo, permitiendo al estudiante interactuar con su entorno inmediato.

Algunos docentes han combinado el aprendizaje de los fundamentos de programación con procesos de realidad mixta mediante un entorno natural, buscando la adquisición de aprendizaje significativo [53].

Guerreo et al [112] clasifica en cuatro categorías a las herramientas utilizadas para la enseñanza-aprendizaje de la programación: Herramientas de calificación automática. 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 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.

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 del LMS de programación y herramientas de software que mediante videos y screencast ayudan al estudiante en la búsqueda de ideas para resolver problemas de programación [113]; utilización de imágenes intuitivas que visualicen paso a paso la ejecución de cada instrucción de un código fuente [114]; la incorporación de un diario en el LMS como registro del proceso llevado a efecto durante el tiempo de aprendizaje [115].

Sistemas inteligentes de tutoría 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. 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 aprendizaje de programación. Entre ellas se encuentran: Logo, Robot Karel20, JKarel Robot, Turingal, Scratch, Greenfoot, Alice, PLM, Robot Scribbler.

4.4 Consideraciones metodológicas en un primer curso de programación de computadores

La revisión sistemática también las siguientes consideraciones metodológicas las cuales se han categorizado como lo muestra la Tabla 13.

Por su parte, Romero y Rosero [47] determinan que, a lo largo de la historia, en la enseñanza de la programación han convivido diversos enfoques y tendencias [116] y que en la actualidad no hay una decisión única en la selección de métodos

de enseñanza ni en los enfoques didácticos a utilizar.

En este sentido, Kaasbøll [117] plantea tres modelos didácticos clásicos en la enseñanza de la programación:

—Escalera semiótica: se basa en la utilización de lenguajes como partida de los procesos de enseñanza de los fundamentos de programación, bajo una concepción de secuencias sintácticas, semánticas y pragmáticas.

—Objetivos de taxonomía cognitiva: se basa en la taxonomía por objetivos de Bloom y se enfoca en el uso de instrucciones para cimentar el desarrollo algorítmico en una actividad.

—Resolución de problemas: es un clásico modelo de aprendizaje ampliamente validado y reconocido por la comunidad académica y científica.

Por su parte, Silva *et al.* [26] proponen dos elementos a considerar en el proceso de enseñanza-aprendizaje de la programación de computadores:

—Enfoque pedagógico: existen diversos enfoques en la enseñanza de los fundamentos de programación como el paradigma de programación, en el lenguaje utilizado, el uso de herramientas de simulación y la visualización, entre otros.

—Factores del estudiante en el aprendizaje de la programación: entre las características estudiantiles más analizadas están: el género, los antecedentes matemáticos, la experiencia previa en programación y la motivación.

Rodríguez [45] sugiere la siguiente clasificación para la enseñanza de los fundamentos de programación, de acuerdo al componente lúdico:

Tabla 13. Categorización de metodologías para un primer curso de programación
Fuente: elaboración propia.

Hallazgo de categoría	Modelos didácticos	Enfoques pedagógicos	Componente lúdico	Estilos de aprendizaje	Estructura curricular
Escalera semiótica	X				X
Objetivos de taxonomía cognitiva	X				X
Resolución de problemas	X	X			
Didácticas propias	X			X	X
Constructivista		X			X
Construccionista		X			X
Basado en metodología del proceso didáctico		X			X
Basados en la enseñanza		X			X
Basados en el currículo		X			X
Desarrollo instruccional basado en vídeo juegos	X		X		X
Desarrollo instruccional orientado hacia el paradigma	X		X		X
Desarrollo instruccional enfocado en temáticas particulares			X		
VARK	X			X	X
Basadas en codificar y depurar	X				X
Implementación de algoritmos básicos	X				X
Enfocadas a la complejidad algorítmica	X				X

—Desarrollo instruccional orientado hacia el paradigma: Dr. Java y Bluej son los proyectos más representativos de este enfoque.

—Desarrollo instruccional basado en videojuegos: Scratch es el mayor representante de este enfoque; además, también se encuentran proyectos como Greenfoot y Alice, que han logrado un avance significativo en la reducción del abandono o pérdida de los cursos de programación que los contemplan.

—Desarrollo instruccional enfocado en temáticas particulares: este enfoque combina una herramienta de programación existente con una metodología de desarrollo de software como XP (eXtreme Programming), ADRI (Approach Deployment Result Improvement), etc.

Estos proyectos están orientados al desarrollo de una característica o propuesta sugerida por el investigador y, en general, integran herramientas existentes con metodologías.

Por su parte, Olague *et al.* [34] determina que uno de los estilos de aprendizaje más utilizado por los estudiantes de un primer curso de programación es el modelo VARK: kinestésico-auditivo, visual-kinestésico-lectoescritura y kinestésico-auditivo-visual lectoescritura, que actúa como la combinación de los estilos anteriores. Como resultado propone establecer las diversas estrategias de enseñanza mediante acciones de predominancia práctica sobre las teóricas; asimismo, los ejercicios planteados deben tener significancia en el aspecto vivencial del estudiante. Los autores también afirman que el constructivismo es el que se adecua mejor a los principios del enfoque kinestésico y que es necesario que en el proceso de enseñanza de los fundamentos de programación se incorporen estrategias de explicación visual, auditiva y escrita, para que el estudiante cuente con los tres estilos de aprendizaje y pueda elegir el que más se ajuste a sus necesidades de aprendizaje.

De acuerdo a las metodologías de enseñanza aplicadas en los cursos de programación, Ortega *et al.* [4] concluyen que se puede determinar que el punto de vista no solo es constructivista, sino construccionista, es decir, se asume que los alumnos aprenden según construyen artefactos, en este caso programas o algoritmos. Adicionalmente, determinan que se ha desarrollado una gran variedad y número de sistemas para el aprendizaje de la programación en niveles universitarios y de forma muy simplificada, entre los cuales se pueden distinguir dos clases [118]: (1) la mayor parte de los sistemas usados para aprender a programar ayudan a comprender la mecánica de la Programación, bien de forma estática o dinámica; (2) otros sistemas facilitan que el alumno aprenda a programar mientras hace alguna tarea que le interesa como jugar.

Sáez *et al.* [30] sugieren tres tipos de problemas para implementar algoritmos:

—Problemas tipo uno: el estudiante solo se enfoca en acciones de codificación y depuración de sus ejercicios planteados y puede incluir la asistencia del profesor para afianzar el proceso de descubrimiento de sus habilidades de programación.

—Problemas tipo dos: para la implementación de los algoritmos, el estudiante se enfoca en cuatro acciones: analizar el problema, diseñar el algoritmo, codificar y depurar la propuesta de solución con ejercicios que admitan la inserción de estructuras lógicas de mayor complejidad de forma gradual.

—Problemas tipo tres: el estudiante centra su atención en las cuatro acciones ya mencionadas, pero con problemáticas que incorporen complejidad al más alto grado, con el fin de retar cada vez más al estudiante a reformar su estructura mental frente a la solución de problemas.

Por su parte, Hernández *et al.* [5] establecen que los retos que plantea la

didáctica de la programación de computadoras han sido abordados desde tres diferentes enfoques:

—Un primer enfoque se centra en la metodología del proceso didáctico.

—Un segundo enfoque se centra en las formas de enseñanza y plantea didácticas propias.

—Un tercer enfoque se centra en los contenidos para plantear alternativas y así fortalecer las habilidades en los estudiantes.

Finalmente, Insuasti [21] presenta dos sugerencias relacionadas con la evaluación en los resultados obtenidos en un primer curso de programación de computadores, las cuales se pueden clasificar en:

—Sugerencias simples: referidas en el cambio del lenguaje de programación.

—Sugerencias detalladas: relacionadas con los modelos y paradigmas apropiados en la enseñanza de los lenguajes de programación.

En este punto es apropiado formular algunas recomendaciones para afrontar un primer curso de programación de computadores, de acuerdo a los resultados obtenidos en la revisión sistemática de la literatura.

Los dos enfoques de mayor utilización en la enseñanza de los fundamentos de programación son el enfoque de objetos y el de programación estructurada [21].

Sin embargo, al tratar de estudiarlos en cursos continuos y en este orden, Sheard y Hagan [119] establecieron que los estudiantes se enfrentaban a un choque conceptual que les generaba confusión.

En consecuencia, se recomienda comenzar con el enfoque estructurado (*bottom-up*) y luego con la orientación a objetos. Esta propuesta fue aceptada por muchas instituciones en su componente curricular y derivó en la obtención de mejores resultados académicos en los estudiantes [119].

Por otra parte, Herbert [120] concluye que, al reducir elementos en la sintaxis de un código, incluir explicaciones visuales en el manejo de sus líneas y brindar las mejores alternativas de solución, es posible mejorar el aprendizaje de los fundamentos de programación [21].

Por su parte, Dann, Cooper y Pausch [29], [21] establecen que el desarrollo de pensamiento algorítmico, el manejo de la abstracción y la percepción en detalle de la realidad son las competencias que un estudiante de un primer curso de programación debe afianzar [21].

Así mismo, Ali [121] recomienda incluir en los ejemplos de clase de los cursos introductorios de programación problemas cotidianos que estén en el campo de dominio de los estudiantes, con el objetivo de obtener mejores niveles de abstracción.

Silva *et al.* [26] sugieren dos opciones pedagógicas para mejorar los procesos tanto de enseñanza como de aprendizaje de un primer curso de programación: la inclusión de un curso de preprogramación y el uso de un tipo específico de herramienta de apoyo a la programación [26]. De hecho, la introducción de un curso de preprogramación, denominado también CS0, demostró que los estudiantes que tomaron aquellos cursos en los que se enseñaba la resolución de problemas, el desarrollo de algoritmos, la generación de pseudocódigos y la diagramación, podían usar un pseudocódigo de forma más consistente que otra cohorte [101].

Entonces, el hecho de contar en un currículo con un primer curso de algoritmos y fundamentos de programación es una buena opción que contribuye con las habilidades mínimas necesarias para la solución de problemas computacionales del estudiantado [41].

Aunado a lo anterior, se debe tener en cuenta que el objetivo particular de la enseñanza de la programación debe centrarse en el desarrollo de pensamiento computacional y algorítmico para el desarrollo de problemas, mas no en la mera escritura de secuencias para ejecutarlas en un computador [122].

Para iniciar a estudiantes en la programación, se deben estimar dos situaciones [74]:

—Formación en las universidades: considera la posibilidad de medir los niveles de abstracción en los estudiantes de un primer curso de programación.

—Mecanismos de selección: permitirían evaluar habilidades de abstracción antes de inscribirse en la universidad.

Rodríguez [45] afirma que, en la actualidad, el lenguaje de programación Python es uno de los protagonistas en la enseñanza de la programación [123] y tiene su mayor aceptación en muchos centros de estudio de Estados Unidos.

4.5 Tendencias de la programación de computadores

Trejos [9] propone nuevas tendencias de la programación en un futuro cercano, entre las cuales destacan:

—La programación literaria: pretende excluir al código fuente entre el programador y la máquina, para dejar únicamente soluciones mediadas por lenguajes naturales.

—La programación inteligente: busca obtener programas de cómputo que se actualicen automáticamente, según las necesidades de los usuarios y sin la intervención de un programador [124].

—La programación virtual: esta tendencia aspira a que el software no se limite únicamente a sentencias de código, sino que desarrolle soluciones que den respuestas correctas a situaciones específicas. Aunque en la actualidad esta tendencia está en teoría, no se descarta la posibilidad de implementarla en un futuro próximo.

—La programación para la animación: ese tipo de programación busca duplicar la realidad con el más mínimo detalle en entornos virtuales, lo que le permite al usuario interacciones como aquellas que desarrolla en su vida real.

Asimismo, existen otras tendencias que por el momento parecen distantes, pero pueden convertirse en realidad en un futuro no muy lejano, entre ellas están:

—Aprendizaje Basado en Investigación (ABI): las nuevas tendencias autodidactas proponen nuevos escenarios para el aprendizaje de la programación, enmarcados en procesos investigativos de alto nivel que los estudiantes de educación superior puedan llegar a desarrollar, con el objeto de profundizar y ampliar los conocimientos iniciales en forma exponencial.

—Programación infantil: aunque hoy en día muchos países han integrado cursos para el desarrollo del pensamiento computacional en sus currículos de educación primaria y secundaria, sin el propósito de formar desarrolladores de software, sino como un nuevo enfoque para solucionar problemas de la vida real, existe la posibilidad de contemplar cursos completos de programación de computadores a temprana edad en niños en etapa escolar, debido a su gran potencial para aprender fácilmente y desarrollar software.

—La programación 3D: puede llegar a convertirse en un paradigma formal y, a pesar de que, por el momento, solo es una proyección desde contextos bidimensionales, se espera el desarrollo del hardware adecuado para su potencialización.

—Programación natural: en un futuro, los programadores ya no se preocuparán por escribir sus propias líneas de código mediante lenguajes de programación formales, pues contarán con contextos enriquecidos por el reconocimiento de audio, video y procesamiento inteligentes.

Estos, aparte de incorporar líneas de código mediante comandos de voz, se encargarán de todos los procesos requeridos en la construcción, validación, documentación y actualización de software de calidad.

—Programación por etapa de vida: llegará un momento en que la gran

mayoría de las personas tendrán conocimientos de programación y, según su etapa de vida (niñez, adolescencia, juventud, adultez, ancianidad), se dedicarán a construir aplicaciones en beneficio de sus propias necesidades.

5. DISCUSIÓN

Respecto a las dificultades en el proceso de aprendizaje de la programación, la gran mayoría de autores consultados en la revisión sistemática 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) de un primer curso de programación de computadores. Sin embargo, también encuentran relevantes algunos inconvenientes relacionados con los estilos de aprendizaje, las limitaciones en su capacidad de abstracción, la cantidad de tiempo —escaso para asimilar cantidades enormes de conceptos nuevos— y los continuos cambios en los entornos de programación.

Asimismo, la investigación arroja que, a pesar de que la enseñanza de la programación 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 en que se debe afrontar un primer curso de programación, especialmente, 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.

Los estudios revelan que, en la orientación del primer curso de programación, aún se sigue el modelo por imitación de la escuela tradicional, en el cual la didáctica gira en torno a la explicación por parte del profesor, quien todavía guarda la esperanza de que el estudiante descubra los modelos propuestos a través de la solución de ejercicios demostrativos, en los no se prevén los procesos introspectivos de planeación, control y evaluación con el rigor adecuado.

De igual manera, en la enseñanza de la programación de computadores confluye el desarrollo de muchas habilidades como la resolución de problemas, el modelado de situaciones o problemas, la incorporación de la eficiencia en soluciones, el dominio de uno o varios lenguajes de programación, entre otras. Estas hacen más complejo el objetivo de asumir y cumplir el propósito fundamental de dicho curso.

Acerca de las herramientas reportadas para asumir un primer curso de programación de computadores, se puede apreciar que existen varios esfuerzos para facilitar el aprendizaje al estudiante novel, que van desde herramientas de visualización de algoritmos y programas hasta la adaptación de tecnologías emergentes como en el caso de la realidad aumentada y la realidad mixta, para apoyar los procesos de abstracción. Además, se puede apreciar que el campo de los juegos tanto en su creación como en su utilización en un primer curso despierta una motivación adicional en el estudiante.

Por otra parte, se observa una interesante cuantía de estrategias de enseñanza-aprendizaje adaptadas y desarrolladas para guiar en el proceso tanto a estudiantes como a docentes, en las cuales la revisión sistemática registra un equilibrio entre la cantidad de estrategias centradas en el aprendizaje frente a las centradas en la enseñanza. Aunado a esto, se evidencia que las estrategias colaborativas hacen un importante aporte al proceso de enseñanza-aprendizaje al incorporar técnicas de colaboración y cooperación en los estudiantes. Asimismo, se debe resaltar la incorporación de estrategias clásicas que van desde las clases magistrales y el laboratorio hasta las

utilizadas en otras áreas del saber cómo las analogías y las metáforas.

En cuanto a las consideraciones metodológicas, se puede establecer que no existe un consenso en métodos de enseñanza ni en procesos de aprendizaje establecidos, pese al momento e importancia del software en la actualidad. Igualmente, existe una importante cantidad de propuestas de modelos didácticos, enfoques pedagógicos, estilos de aprendizajes y metodologías de enseñanza propias y adaptadas, que establecen algunos lineamientos para los docentes nuevos o aquellos que buscan perfeccionar su metodología de enseñanza, para abordar un primer curso de programación de computadores, con grupos de estudiantes que varían en expectativas y capacidades en cada periodo académico.

Por su parte, en la revisión sistemática se hallaron importantes recomendaciones para afrontar un primer curso de programación, muchas de las cuales sugieren fortalecer los niveles de abstracción mediante el desarrollo del pensamiento algorítmico. Así mismo, se espera que estos puedan ser potenciados con herramientas de visualización y la utilización de un lenguaje de programación que minimice su sintaxis de codificación, gracias al uso de ejemplos inicialmente relacionados con experiencias ya adquiridas por los estudiantes. Tal es el caso de la física, la química, la matemática, la geometría y la estadística, entre otras disciplinas, que pueden ser combinadas con sucesos cotidianos cercanos a sus experiencias (por ejemplo, el sistema cuantitativo de notas, las compras, la facturación, etc.), para luego formular pequeños proyectos, en los que puedan adoptar algunas de las herramientas, metodologías y estrategias individuales y colectivas citadas anteriormente.

Finalmente, la revisión sistemática genera un hallazgo de las futuras tendencias que tiene la programación de computadores, las cuales se orientan al tratamiento del diseño de software para generación y adaptación de código automático e inteligente, con tendencias hacia la virtualidad, animación y proyección 3D.

6. CONCLUSIONES

Hoy en día son evidentes los esfuerzos de docentes e investigadores enfocados en el estudio de los procesos de enseñanza-aprendizaje para un primer curso de programación, debido a su importancia e incidencia en el desarrollo de las habilidades de los estudiantes tanto en su vida académica como en el futuro desarrollo profesional en la construcción de software.

Por este motivo, para este artículo se adelantó un método de revisión sistemática de literatura (sección 3), que permitió recopilar, sintetizar y categorizar los hallazgos de experiencias relacionadas con los procesos de enseñanza-aprendizaje en un primer curso de programación de computadores, reportado en la educación superior.

Como resultado, se obtuvo un total de 106 estudios en cuatro bases de datos de publicaciones científicas de los últimos siete años, que incluyeron criterios de búsqueda de término principal con sus correspondientes sinónimos y filtros adicionales (sección 3.2), de los cuales, quince fueron duplicados y 41 excluidos previa aplicación de la evaluación de calidad (sección 3.4), de acuerdo a los criterios de selección establecidos (sección 3.3). En esa medida, los 50 estudios restantes permitieron dar respuesta a las preguntas de investigación formuladas en esta revisión sistemática.

Los hallazgos reportados en la sección 2 responden a las preguntas RQ1 y RQ2 y permiten determinar que existen varios problemas relacionados con los procesos de enseñanza/aprendizaje tanto por parte de los estudiantes como de los docentes a cargo de los cursos. Dichos problemas incluyen

inconvenientes en los estilos de aprendizaje, falta de experiencias previas, nivel de abstracción en los conceptos, tiempo de estudio, cantidad de conceptos, inadecuadas técnicas de estudio, inexistencia consensos en torno a las herramientas, metodologías, didácticas, saberes, competencias, modelos y demás elementos necesarios para afrontar un primer curso de programación.

En la Tabla 1 se resumen las 33 herramientas que dan respuesta a la pregunta RQ3, explicadas por extensión en la sección 4.1, categorizadas en herramientas de visualización o simuladores de algoritmos, evaluación automática, juegos educativos centrados en la enseñanza de una unidad específica de aprendizaje, juegos educativos centrados en la enseñanza de unidades múltiples de aprendizaje y ambientes colaborativos.

En la Tabla 2, se categorizan 36 estrategias para un primer curso de programación, que dan respuesta a la pregunta RQ4 y clasificadas en colaborativas, centradas en el aprendizaje y centradas en la enseñanza, las cuales se detallan en la sección 4.2.

Como respuesta a la pregunta RQ5, hay 18 consideraciones metodológicas vinculadas a modelos didácticos, enfoques pedagógicos, componente lúdico, estilos de aprendizaje y estructura curricular.

Están resumidas en la Tabla 3 y descritas en la sección 4.3. Además, entre los hallazgos de la revisión sistemática, hay una interesante cantidad de recomendaciones para afrontar un primer curso de programación, las cuales responden a la pregunta inicial y establecen al menos un punto de partida a tener en cuenta tanto para los docentes novatos, en la orientación de un primer curso, como para aquellos que ya han tenido experiencia y buscan nuevas alternativas para mejorar los resultados obtenidos hasta el momento.

Dichas recomendaciones se presentan con detalle en la sección 4.4. Finalmente, ante la pegunta RQ6, se encontró una categorización de las posibles tendencias de la programación de computadores que se amplían en la sección 4.5 y se enmarca en el tratamiento de código fuente, su adaptación y su tendencia hacia el desarrollo emergente tecnológico.

7. AGRADECIMIENTOS

Expresamos nuestro agradecimiento a los integrantes del Grupo de investigación Tecnofilia, de la Facultad de Ingeniería de la Universidad Cesmag; al Grupo de Investigación y Desarrollo en Ingeniería del Software (IDIS) de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca y al Grupo de Investigación Galeras.net del Departamento de Sistemas de la Universidad de Nariño.

8. REFERENCIAS

[1] C. Malliarakis, M. Satratzemi, y S. Xinogalos, "Educational Games for Teaching Computer Programming," en *Research on e-Learning and ICT in Education*, New York: Springer, 2014, pp. 87–98. https://doi.org/10.1007/978-1-4614-6501-0_7

[2] K. M. Y. Law, V. C. S. Lee, y Y. T. Yu, "Learning motivation in e-learning facilitated computer programming courses," *Comput. Educ.*, vol. 55, no. 1, pp. 218–228, Aug. 2010. https://doi.org/10.1016/j.compedu.2010.01.007

[3] S. Fincher y M. Petre, *Computer Science Education Research*, London, UK: Taylor and francis Group, 2004. Disponible en: URL

[4] M. Ortega *et al.*, "iProg: Development of immersive systems for the learning of programming," en *Proceedings of the XVIII International Conference on Human Computer Interaction - Interacción* Cancún, 2017, pp. 1–6. https://doi.org/10.1145/3123818.3123874

[5] G. Hernández- Pantoja, "Creencias docentes y didáctica de la programación de computadoras," *Docencia investigación innovación*, vol. 2, no. 2, pp. 87–103, Dic. 2013. Disponible en: URL

[6] M. O. Galdeano y F. G. O. Uribe, "La enseñanza de la programación," *Academias de computación de la UPIICSA*. 2002. Disponible en: URL

[7] M. Juganaru, *Inroducción a la programación*, 1st ed. México: Patria S.A. de C.V, 2014. Disponible en: URL

[8] N. Couthinjo, *Introducción a la programación con Python: Algoritmos y lógica de programación para principiantes*, 1st ed. Sao Paulo, Brasil: Novatec, 2016. Disponible en: URL

[9] O. Trejos-Buriticá, "Consideraciones sobre la evolución del pensamiento a partir de los paradigmas de programación de computadores," *Tecnura*, vol. 16, no. 32, pp. 68–83, Apr. 2012. Disponible en: URL

[10] J. Villalobos y R. Casallas, *Fundamentos de programación: aprendizaje activo basado en casos*. Colombia: Universidad de los Andes, Facultad de ingeniería, 2010. Disponible en: URL

[11] M. Cedano-Olvera, A. Cedano- Rodríguez, J. Rubiano-González, y A. Vega-Gutiérrez, *Fundamentos de computación para ingenieros*. 1st ed. Mexico: Grupo Editorial Patria, 2014. Disponible en: URL

[12] K. Petersen, R. Feldt, S. Mujtaba, y M. Mattsson, "Systematic mapping studies in software engineering" en *EASE'08 Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, Italy, 2008. Disponible en: URL

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

[14] B. A. Kitchenham, T. Dyba, y M. Jorgensen, "Evidence-based software engineering," en *Proceedings. 26th International Conference on Software Engineering*, 2004, pp. 273–281. https://doi.org/10.1109/ICSE.2004.1317449

[15] B. Kitchenham y S. Charters, "Guidelines for performing Systematic Literature reviews in Software Engineering Versión 2.3," Keel University and University of Durham, Technical Report, EBSE-2007-01, Jul. 2007. Disponible en: URL

[16] B. Kitchenham et al., "Systematic literature reviews in software engineering – A tertiary study," *Inf. Softw. Technol.,* vol. 52, no. 8, pp. 792–805, 2010. https://doi.org/10.1016/j.infsof.2010.03.006

[17] O. Revelo-Sánchez, C. A. Collazos-Ordóñez, y J. A. 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, Jan. 2018. https://doi.org/10.22430/22565337.731

[18] J. D. Martínez Díaz, V. Ortega Chacón, y F. J. Muñoz Ronda, "El diseño de preguntas clínicas en la práctica basada en la evidencia. Modelos de formulación," *Enfermería Glob.*, vol. 15, no. 3, pp. 431-438, Jun. 2016. https://doi.org/10.6018/eglobal.15.3.239221

[19] B. O. Depetris, D. Aguil Mallea, H. Pendenti, G. Tejero y G. E. Feierherd, "Experiencias con Da Vinci Concurrente en la enseñanza inicial de la programación y la programación concurrente," en *Proceedings del VIII Congreso de Tecnología en Educación y Educación en Tecnología*, Santiago del Estero, 2013. Disponible en: URL

[20] J. López Reguera, C. Hernández Rivas, y Y. Farran Leiva, "Una plataforma de evaluación automática con una metodología efectiva para la enseñanza/aprendizaje en programación de computadores," *Ingeniare. Rev. Chil. Ing.*, vol. 19, no. 2, pp. 265–277, Aug. 2011. https://doi.org/10.4067/S0718-33052011000200011

[21] J. Insuasti, "Problemas de enseñanza y aprendizaje de los fundamentos de programación" *Educ. y Desarro. Soc.*, vol. 10, no. 2, pp. 234–246, 2016. Disponible en: URL

[22] R. Muñoz, M. Barría, R. Noel, E. Providel, y P. Quiroz, "Determinando las dificultades en el aprendizaje de la primera asignatura de programación en estudiantes de ingeniería civil informática," en *Memorias del XVII Congreso Internacional de Informática Educativa, TISE*, Santiago, 2012, pp. 120–126. Disponible en: URL

[23] A. Carbone, J. Hurst, I. Mitchell, y D. Gunstone, "An exploration of internal factors influencing student learning of programming," en *ACE '09 Proceedings of the Eleventh Australasian Conference on Computing Education*, Wellington, New Zealand, 2009, pp. 25-34. Disponible en: URL

[24] R. M. Felder y R. Brent, "Understanding Student Differences," *J. Eng. Educ.*, vol. 94, no. 1, pp. 57–72, Jan. 2005. https://doi.org/10.1002/j.2168-9830.2005.tb00829.x

[25] S. Casas y V. Vanoli, "Programación y Algoritmos: Análisis y Evaluación de Cursos Introductorios," en *IX Workshop de Investigadores en Ciencias de la Computación*, Comodoro Rivadavia, 2007, pp. 760–764. Disponible en: URL

[26] G. Silva-Maceda, P. D. Arjona-Villicana, y F. E. Castillo-Barrera, "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, Nov. 2016. https://doi.org/10.1109/TE.2016.2535207

[27] C. Watson y F. W. B. Li, "Failure rates in introductory programming revisited," en *Proceedings of the 2014 conference on*

Innovation & technology en computer science education - ITiCSE '14, Uppsala, 2014, pp. 39–44. https://doi.org/10.1145/2591708.2591749

[28] National Science Board, *Science and engineering indicators 2012*. Arlington, VA, Washington: Arlington VA: National Science Foundation (NSB 12-01), 2012, pp. 589. Disponible en: URL

[29] W. Dann, S. Copper, and R. Pausch, *Learning to Program with Alice (w/ CD ROM)*, 3rd ed., New York: Prentice Hall Press Upper Saddle River, 2011. Disponible en: URL

[30] A. Saez Villavicencio, F. A. Ciudad Ricardo, U. Puentes- Puentes, y J. Menéndez- Pérez, "El desarrollo de la habilidad : implementar algoritmos . Teoría para su operacionalización," *Rev. Cuba. Ciencias Informáticas*, vol. 9, no. 3, pp. 99–112, Jul. 2015. Disponible en: URL

[31] M. Bozorgmanesh, M. Sadighi, y M. Nazarpour, "Increase the eficiency of adult education with the proper use of learning styles," *Nat. Sci.*, vol. 9, no. 5, pp. 140–145, Jun. 2011. Disponible en: URL

[32] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," *ITiCSE '05 Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, Caparica, 2005, no. 3, pp. 14-18. https://doi.org/10.1145/1151954.1067453

[33] L. P. Baldwin y J. Kuljis, "Learning programming using program visualization techniques," en *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, Maui, 2001. https://doi.org/10.1109/HICSS.2001.926232

[34] J. R. Olague-Sánchez, *et al.*, "Sistemas de gestión de contenidos de aprendizaje y técnicas de minería de datos para la enseñanza de ciencias computacionales, un caso de estudio en el norte de Coahuila," *Rev. Mex. Investig. Educ.*, vol. 15, no. 45, pp. 391–421, Apr. 2010. Disponible en: URL

[35] E. Dunican, "Making the analogy: Alternative delivery techniques for first year programming courses" en *Proceedings from the 14th Workshop of the Psychology of Programming Interest Group, Brunel University*, 2002, pp. 89–99. Disponible en: URL

[36] B. Depetris, D. Aguil Mallea, H. Pendenti, G. Tejero y G. E. Feierherd, "Experiencias con Da Vinci Concurrente en la enseñanza inicial de la programación y la programación concurrente," en *Proceedings del VIII Congreso de Tecnología en Educación y Educación en Tecnología*, La Plata, 2013. Disponible en: URL

[37] L. Thomas, M. Ratcliffe, J. Woodbury, y E. Jarman, "Learning styles and performance in the introductory programming sequence," en *Proceedings of the 33rd SIGCSE technical symposium on Computer science education - SIGCSE '02*, Cincinnati, 2002, pp.33-37. https://doi.org/10.1145/563340.563352

[38] I. Stamouli, E. Doyle, y M. Huggard, "Establishing structured support for programming students," en *34th Annual Frontiers in Education, 2004. FIE 2004.*, Savannah, 2004, pp. 679–683. https://doi.org/10.1109/FIE.2004.1408612

[39] W. Hartmann, J. Nievergelt, y R. Reichert, "Kara, finite state machines, and the case for programming as part of general education," en *Proceedings IEEE Symposia on Human-Centric Computing Languages and Environments (Cat. No.01TH8587)*, Stresa, 2001, pp. 135–141. https://doi.org/10.1109/HCC.2001.995251

[40] V. Frittelli *et al.*, "Desarrollo de juegos como estrategia didáctica en la enseñanza de la programación," *CONAIIS*I, 2013. Disponible en: URL

[41] J. Sánchez García, M. Urías Ruiz, y B. Gutiérrez Herrera, "Análisis de los problemas de aprendizaje de la Programación Orientada a Objetos," *Ra Ximhai*, vol. 11, no. 4, pp. 289–304, Jul. 2015. Disponible en: URL

[42] L. Spigariol y N. Passerini, "Enseñando a programar en la orientación a objetos," en *Congreso Nacional de Ingeniería Informática / Sistemas de Información. Educación en Ingeniería.*, Buenos Aires, 2013, vol. 1. Disponible en: URL

[43] L. López- Román, "Metodología para el desarrollo de la lógica de la programación orientada a objetos," *Sist. Cibernética e Informática*, vol. 10, no. 2, pp. 27–32, 2013. Disponible en: URL

[44] M. Sicilia, "Beyond content: sharing the design of open educational resources," *RUSC. Univ. Knowl. Soc. J.*, vol. 4, no. 1, pp. 1580–1698, Apr. 2007. https://doi.org/10.7238/rusc.v4i1.297

[45] 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. https://doi.org/10.26620/uniminuto.inventum.9.17.2014.51-61

[46] O. I. Trejos- Buriticá , "Modelo de enseñanza con aprendizaje colaborativo en estudiantes de programación de computadores," *Rev. Vector*, vol. 9, pp. 48–58, Aug. 2014. Disponible en: URL

[47] C. Romero-Chaves y M. Rosero-Sosa, "Modelo de enseñanza y su relación con los procesos

metacognitivos en programación de sistemas," *Rev. Educ. en Ing.*, vol. 9, no. 17, pp. 1–12, Jun. 2014. Disponible en: URL

[48] J. J. Arellano-Pimentel, O. S Nieva- García, R. Solar Gonzáez, y 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, Dic. 2012. Disponible en: URL

[49] T. Crews y J. Butterfield, "Using technology to bring abstract concepts into focus: A programming case study," *J. Comput. High. Educ.*, vol. 13, no. 2, pp. 25–50, Mar. 2002. https://doi.org/10.1007/BF02940964

[50] M. C. Carlisle, T. A. Wilson, J. W. Humphries, y S. M. Hadfield, "Raptor: A visual programming environment for teaching algorithmic problem solving," en *Proceedings of the 36th SIGCSE technical symposium on Computer science education (SIGCSE' 05)*, New York, 2005, pp. 176-180. Disponible en: URL

[51] A. Del Prado y N. Lamas, "Alternativas para la enseñanza de pseudocódigo y diagrama de flujo," *Rev. Electrónica Iberoam. Educ. en Ciencias y Tecnol.*, vol. 5, no. 3, pp. 102–113, Dic. 2014. Disponible en: URL

[52] Jeliot 3 (2018). Disponible en: URL

[53] S. Sánchez *et al.*, "Applying Mixed Reality Techniques for the Visualization of Programs and Algorithms in a Programming Learning Environment," en *nternational Conference on Mobile, Hybrid, and On-line Learning*, Rome, 2018, pp. 84–89. Disponible en: URL

[54] K. A. Reek, "The TRY System or how to Avoid testing students programs," en *Proceedings of the twentieth SIGCSE technical symposium on Computer science education*, Louisville, Kentucky, 1989, pp. 112–116. https://doi.org/https://doi.org/10.1145/65294.71198

[55] E. L. Jones, "Grading student programs a software testing approach," *J. Comput. Sci. Coll.*, vol. 16, no. 2, pp. 185-192, Jan. 2001. Disponible en: URL

[56] L. Malmi, V. Karavirta, A. Korhonen, y J. Nikander, "Experiences on automatically assessed algorithm simulation exercises with different resubmission policies," *J. Educ. Resour. Comput.*, vol. 5, no. 3, Sep. 2005. https://doi.org/10.1145/1163405.1163412

[57] C. Higgins, G. Gray, P. Symeonidis, y A. Tsintfsias, "Automated assessment and experiences of teaching programming," *J. Educ. Resour. Comput.*, vol. 5, no. 3, pp. 1–21, Sep. 2005. https://doi.org/https://doi.org/10.1145/1163405.1163410

[58] T. Wang, X. Su, P. Ma, Y. Wang, y K. Wang, "Ability-training-oriented automated assessment in introductory programming course," *Comput. Educ.*, vol. 56, no. 1, pp. 220–226, Jan. 2011. https://doi.org/10.1016/j.compedu.2010.08.003

[59] T. Barnes, A. Chaffinet, A. Godwin, E. Powell, H. Richter "The Role of Feedback," 2007, pp. 1–5. Disponible en: URL

[60] T. Barnes, A. Chaffin, E. Powell, y H. Lipford, "Game2Learn: Improving the motivation of CS1 students," en *Proceedings of the 3rd international conference on Game development in computer science education*, Miami, 2008, pp. 1–5. https://doi.org/10.1145/1463673.1463674

[61] A. Chaffin, K. Doran, D. Hicks, y T. Barnes, "Experimental evaluation of teaching recursion in a video game," en *Proceedings ACM SIGGRAPH Symposium on Video Games*, New Orleans, Louisiana, 2009, pp. 79–86. Disponible en: URL

[62] M. Eagle y T. Barnes, "Experimental evaluation of an educational game for improved learning in introductory computing," en *Proceedings of the 40th ACM technical symposium on Computer science education - SIGCSE '09*, Chattanooga, 2009, pp. 321-325. https://doi.org/10.1145/1508865.1508980

[63] F. W. B. Li y C. Watson, "Game-based concept visualization for learning programming," en *Proceedings of the third international ACM workshop on Multimedia technologies for distance learning - MTDL '11*, Scottsdale, Arizona, 2011, pp. 37-42. https://doi.org/10.1145/2072598.2072607

[64] M. Piteira y S. R. Haddad, "Innovate in your program computer class: an approach based on a serious game" en *Proceedings of the 2011 Workshop on Open Source and Design of Communication - OSDOC '11*, Lisboa, 2011, pp. 49-54. https://doi.org/10.1145/2016716.2016730

[65] K. Maragos y M. Grigoriadou, "Exploiting Talent as a Tool for Teaching and Learning," *Int. J. Learn.*, vol. 18, no. 1, pp. 431–439, Jan. 2011. Disponible en: URL

[66] M. J. Lee y A. J. Ko, "Personifying programming tool feedback improves novice programmers' learning," en *Proceedings of the seventh international workshop on Computing education research - ICER '11*, Providence, Rhode Island, 2011, pp. 109-116. https://doi.org/10.1145/2016911.2016934

[67] C. Palma- Suárez y R. Sarmiento- Porras, "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, Apr. 2015. Disponible en: URL

[68] J. O'Kelly y J. P. Gibson, "RoboCode & problem-based learning: a non-prescriptive approach to teaching programming," *ACM SIGCSE Bull.*, vol. 38, no. 3, pp. 217-221, Jun. 2006. https://doi.org/10.1145/1140123.1140182

[69] A. M. Phelps, C. A. Egert, y K. J. Bierre, "MUPPETS: Multi-User Programming Pedagogy for Enhancing Traditional Study: An Environment for both Upper and Lower Division Students," en *Proceedings Frontiers in Education 35th Annual Conference*, Indianopolis, 2004, pp. S2H-8-S2H-15. https://doi.org/10.1109/FIE.2005.1612247

[70] M. Muratet, P. Torguet, F. Viallet, y J. P. Jessel, "Experimental Feedback on Prog&Play: A Serious Game for Programming Practice," *Comput. Graph. Forum*, vol. 30, no. 1, pp. 61–73, Oct. 2010. https://doi.org/10.1111/j.1467-8659.2010.01829.x

[71] I. Paliokas, C. Arapidis, y M. Mpimpitsos, "PlayLOGO 3D: A 3D Interactive Video Game for Early Programming Education: Let LOGO Be a Game," en *2011 Third International Conference on Games and Virtual Worlds for Serious Applications*, Athens, 2011, pp. 24–31. https://doi.org/10.1109/VS-GAMES.2011.10

[72] M. J. Lee y A. J. Ko, "Personifying Programming Tool Feedback Improves Novice Programmers' Learning," en *Proceedings of the seventh international workshop on Computing education research, ICER '11,* Providence, 2011, pp. 109–116. https://dx.doi.org/10.1145/2016911.2016934

[73] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking (2012)," in *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, 2012. Disponible en: URL

[74] R. F. Zúñiga Muñoz, J. A. Hurtado Alegría, y P. Paderewsky Rodríguez, "Discovering the mechanisms of abstraction in the performance of work teams in children to solve computational problems," *Sist. y Telemática*, vol. 14, no. 36, pp. 69–87, Mar. 2016. https://doi.org/10.18046/syt.v14i36.2216

[75] J. Mönig y B. Harvey. "Snap: Build your own blocks," Snap, 2019. Disponible en: URL

[76] C.-C. Liu, Y.-B. Cheng, y C.-W. Huang, "The effect of simulation games on the learning of computational problem solving," *Comput. Educ.*, vol. 57, no. 3, pp. 1907–1918, Nov. 2011. https://doi.org/10.1016/j.compedu.2011.04.002

[77] J. García, P. Señas, y N. Moroni, "Cubik: Una Herramienta de Apoyo a la Enseñanza de la Programación," en *'IV Ateneo de Profesores Universitarios de Computación. San Luis.*, 1996, pp. 582- 594. Disponible en: URL

[78] G. Cenich, "Un entorno para la enseñanza y aprendizaje de programación en la escuela secundaria," *en III Jornadas de TIC e Innovación en el Aula, La Plata, 2015*, pp.1-9. Disponible en: URL

[79] M. Gallego y F. Gortázar, "EclipseGavab, un entorno de desarrollo para la docencia online de la programación," in *Jornadas de Enseñanza Universitaria de la Informática (JENUI) - JENUI 2009 [68].* Barcelona, 2009, pp. 501-508. Disponible en: URL

[80] E. Lovos, A. Gonzalez, I. Mouján, R. Bertone, y C. Madoz, "Estrategias de Enseñanza Colaborativa para un Curso de Programación de Primer Año de la Licenciatura en Sistemas," in *XVIII Congreso Argentino de Ciencias de la Computación*, Bahia Blanca, 2012. Disponible en: URL

[81] J. C. Rodríguez del Pino, E. Rubio-Royo, y Z. Hernández-Figueroa, "VPL: laboratorio virtual de programación para Moodle" en *Jornadas de Enseñanza Universitaria de la Informática (JENUI) - JENUI 2010, Barcelona,* 2010, pp. 429-435. Disponible en: URL

[82] J. A. Jiménez Builes, M. Pavony Meneses, y A. F. Alvarez Serna, "Entorno de integración de PBL y CSCL para la enseñanza de algoritmos y programación en ingeniería," *Rev. Av. en Sist. e Informática*, vol. 5, no. 3, pp. 189--194, Dic. 2008. Disponible en: URL

[83] Grupo Chico, Universidad de la Mancha "Colecce 2.0," ¿What is COLLECE 2.0?, 2018. Disponible en: URL

[84] N. Moroni y P. Señas, "La Visualización de Algoritmos como Recurso para la Enseñanza de la Programación," en *IV Workshop de Investigadores en Ciencias de la Computación. WICC*, La Plata, 2002, pp. 213–217. Disponible en: URL

[85] E. Costelloe, "Teaching Programming. The State of the Art", Dublin, 2004. Disponible en: URL

[86] Z. Li, C. Plaue, y E. Kraemer, "A spirit of camaraderie: The impact of pair programming on retention," en *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*, San Francisco, 2013, pp. 209–218. https://doi.org/10.1109/CSEET.2013.6595252

[87] D. Preston, "Using collaborative learning research to enhance pair programming pedagogy," *ACM SIGITE Newsl.*, vol. 3, no. 1,

pp. 16–21, Jan. 2006. https://doi.org/10.1145/1113378.1113381

[88] N. Salleh, E. Mendes, y 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. https://doi.org/10.1109/TSE.2010.59

[89] G.-J. Hwang, Z.-Y. Liang, y H.-Y. Wang, "An Online Peer Assessment-Based Programming Approach to Improving Students' Programming Knowledge and Skills," en *2016 International Conference on Educational Innovation through Technology (EITT)*, Tainan, 2016, pp. 81–85. https://doi.org/10.1109/EITT.2016.23

[90] J. P. Ucan, O. S. Gomez, y 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. https://doi.org/10.1109/TLA.2016.7587643

[91] C. Patek y 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*, Seattle, 2017, pp. 715–715. https://doi.org/10.1145/3017680.3022445

[92] A. Nylén, N. Thota, A. Eckerdal, P. Kinnunen, M. Butler, y M. Morgan, "Multidimensional analysis of creative coding MOOC forums: a methodological discussion",en *Proceedings of the 15th Koli Calling Conference on Computing Education Research,* Koli, Finland, 2015, pp. 137-141. https://www.doi.org/10.1145/2828959.2828971

[93] A. Robins, J. Rountree, y N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Comput. Sci. Educ.*, vol. 13, no. 2, pp. 137–172, Jun. 2003. https://doi.org/10.1076/csed.13.2.137.14200

[94] A. Pears *et al.*, "A survey of literature on the teaching of introductory programming," *ACM SIGCSE Bull.*, vol. 39, no. 4, pp. 204-223, Dec. 2007. https://doi.org/10.1145/1345375.1345441

[95] S. Wiedenbeck, V. Ramalingam, S. Sarasamma, y 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. https://doi.org/10.1016/S0953-5438(98)00029-0

[96] M. De Raadt, R. Watson, y M. Toleman, "Introductory programming: What's happening today and will there be any students to teach tomorrow?," en *ACE '04 Proceedings of the Sixth Australasian Conference on Computing Education*, Dunedin, 2004, pp. 277–282. Disponible en: URL

[97] A. Dingle and C. Zander, "Assessing the ripple effect of CS1 language choice," *J. Comput. Sci. Coll.*, vol. 16, no. 2, pp. 85–93, Jan. 2001. Disponible en: URL

[98] T. Koulouri, S. Lauria, y R. D. Macredie, "Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches," *ACM Trans. Comput. Educ.*, vol. 14, no. 4, pp. 1–28, Feb. 2015. https://doi.org/10.1145/2662412

[99] L. Mannila y M. de Raadt, "An objective comparison of languages for teaching introductory programming," en *Proceedings of the 6th Baltic Sea conference on Computing education research Koli Calling 2006 - Baltic Sea '06*, Uppsala, 2006. https://doi.org/10.1145/1315803.1315811

[100] S. Davies, J. A. Polack-Wahl, y K. Anewalt, "A snapshot of current practices in teaching the introductory programming sequence," en *Proceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE '11*, Dallas, 2011. https://doi.org/10.1145/1953163.1953339

[101] M. Rizvi, T. Humphries, D. Major, M. Jones, y H. Lauzun, "A CS0 course using Scratch," *J. Comput. Sci. Coll.*, vol. 26, no. 3, pp. 19–27, Jan. 2011. Disponible en: URL

[102] C. Herbert, *An introduction to programming using Alice 2.2*, 2da ed. Boston, Massachusetts: Cengage Learning; 2007. Disponible en: URL

[103] I. Blanchette y K. Dunbar, "How analogies are generated: The roles of structural and superficial similarity," *Mem. Cognit.*, vol. 28, no. 1, pp. 108–124, Jan. 2000. https://doi.org/10.3758/BF03211580

[104] C. A. Collazos, L. Guerrero, and A. Vergara, "Aprendizaje Colaborativo: un cambio en el rol del profesor," en *Proceedings of the 3rd Workshop on Education on Computing*, Punta Arenas, 2001, pp. 1–10. Disponible en: URL

[105] J. D. Dougherty, K. Nagel, A. Decker, y K. Eiselt, "Proceedings of the 45th ACM technical symposium on Computer science education," en *SIGCSE '14 Proceedings of the 45th ACM technical symposium on Computer science education*, Atlanta, Georgia, 2014. Disponible en: URL

[106] R. T. Putnam, D. Sleeman, J. A. Baxter, y L. K. Kuspa, "A Summary of Misconceptions of High School Basic Programmers," *J. Educ. Comput. Res.*, vol. 2, no. 4, pp. 459–472, Nov. 1986.

https://doi.org/10.2190/FGN9-DJ2F-86V8-3FAU

[107] D. Perez-Marin, R. Hijon-Neira, y 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. https://doi.org/10.1109/RITA.2018.2809944

[108] T. Boyle, "Constructivism: A Suitable Pedagogy for Information and Computing Sciences?," 2000. Disponible en: URL

[109] R. Muñoz *et al.*, "Uso de Scratch y Lego Mindstorms como Apoyo a la Docencia en Fundamentos de Programación," en *XXI Jornadas de la Enseñanza Universitaria de Informática*, Andorra La Vella, 2015, pp. 248–254. Disponible en: URL

[110] B. Restrepo Gomez, "Aprendizaje basado en problemas (ABP):una innovación didáctica para la enseñanza universitaria," *Educ. y Educ.*, vol. 8, pp. 9–19, 2005. Disponible en: URL

[111] J. A. Jiménez Toledo, C. A. Collazos Ordoñez, J. A. Hurtado Alegría, y W. L. Pantoja Yépez, "Estrategia colaborativa en entornos tridimensionales como estrategia didáctica de aprendizaje de estructuras iterativas en programación computacional," Rev. *Investigium ire Ciencias Soc. y Humanas*, vol. 6, no. 2, pp. 80–92, Dic. 2015. Disponible en: URL

[112] M. Guerrero, D. S. Guamán, y 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, Feb. 2015. Disponible en: URL

[113] B. San Miguel, S. Aguirre, J. del Alamo, y M. Cortés, "A proposal for enhancing the motivation in students of computer programming," en *Conference name: 5th International Conference of Education, Research and Innovation*, Madrid, 2012, pp. 1157–1164. Disponible en: URL

[114] S. Naz, S. Hamad Shirazi, T. Iqbal, D. Irfan, M. Junaid, y Y. Naseer, "Learning Programming through Multimedia and Dry-run," *Res. J. Appl. Sci. Eng. Technol.*, vol. 7, no. 21, pp. 4455–4463, Jun. 2014. https://doi.org/10.19026/rjaset.7.822

[115] S. Alhazbi, "Using e-journaling to improve self-regulated learning in introductory computer programming course," en *2014 IEEE Global Engineering Education Conference (EDUCON)*, Istanbul, 2014, pp. 352–356. https://doi.org/10.1109/EDUCON.2014.6826116

[116] A. F. Szpiniak y G. A. Rojo, "Enseñanza de la programación," *Rev. Iberoam. Tecnol. en Educ. y Educ. en Tecnol.*, vol. 1, no. 1, pp. 100–109, Dic. 2006. Disponible en: URL

[117] J. Kaasbøll, "Exploring didactic models for programming," *NIK 98–Norwegian Computer Science Conference*, pp. 195–203, Tapir, 1998. Disponible en: URL

[118] C. Kelleher y R. Pausch, "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," *ACM Comput. Surv.*, vol. 37, no. 2, pp. 83–137, Jun. 2005. https://doi.org/10.1145/1089733.1089734

[119] J. Sheard y D. Hagan, "Experiences with teaching object-oriented concepts to introductory programming students using C++," en *Proceedings. Technology of Object-Oriented Languages. TOOLS 24 (Cat. No.97TB100240)*, Beijing, 1997, pp. 310–319. https://doi.org/10.1109/TOOLS.1997.713558

[120] C. Herbert, "An introduction to programming with Alice," 2007. Disponible en: URL

[121] Ali, A., y S. Mensch, "Issues and challenges for selecting a programming language in a technology update course.," *Information Systems Education Journal*, no 7, vol. 85, pp. 1-10. Jul. 2008. Disponible en: URL

[122] MIT, "Learn to Program, Program to Learn," Mit Media Lab, 2013. Disponible en: URL

[123] P. Guo, "Python Is Now the Most Popular Introductory Teaching Language at Top -U.S. -Universities," *Commun. ACM*, 2014. Disponible en: URL

[124] J. Alonso- Jiménez, *Temas de "Programación lógica e I.A."* Open LIbra, 2013. Disponible en: URL