EVIDENCIAS EMPÍRICAS EN EL PROCESO DE ENSEÑANZA DE LA PROGRAMACIÓN

UNA APROXIMACIÓN USANDO ALICE EN EL CONTEXTO UNIVERSITARIO.

MARIUXI GEOVANNA VINUEZA MORALES

DIRIGIDA POR:
DRA. DIANA BORREGO Y DR. DAVID BENAVIDES

TESIS DOCTORAL

Publicado por primera vez en junio de 2022 por el Departamento de Lenguajes y Sistemas Informáticos
ETSI Informática
Avda. de la Reina Mercedes s/n
Sevilla, 41012. ESPAÑA

mvinuezam@unemi.edu.ec

Categorías (ACM 2012):

Categories and subject descriptors:

[500] Applied computing-Computer-managed instruction
[500] Software and its engineering-Empirical software validation
[500] Applied computing-Collaborative learning
[500] Applied computing-Learning management systems
[500] Applied computing-Computer-assisted instruction

General Terms: Design, Theory, Algorithms, Computational thinking.

Additional Key Words and Phrases: Fundamentals of programming and programming language, academic performance, Scratch, Alice, visual programming languages, programming learning and educational programming learning.

Financiación: This work has been partially supported by University of Milagro (UNEMI) with its scholarship program. It has also been partially funded by the Project (RTI2018-101204-B-C22, OPHELIA), funded by: FEDER/Ministry of Science and Innovation - State Research Agency; and the Junta de Andalucia COPERNICA (P20_01224) and METAMORFOSIS (FEDER_US-1381375) projects.

Doña Diana Borrego, profesora del área de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla y Don David Benavides, catedrático del área de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla.

HACEN CONSTAR

que Doña Mariuxi Geovanna Vinueza Morales, Licenciada en Sistemas de Información por la Escuela Superior Politécnica del Litoral, Ecuador, ha realizado bajo nuestra supervisión el trabajo de investigación titulado

Evidencias empíricas en el proceso de enseñanza de la programación.
Una aproximación usando ALICE en el contexto universitario

Una vez revisado, autorizamos el comienzo de los trámites para su presentación como tesis doctoral al tribunal que ha de juzgarlo.

Fdo. Dra. Diana Borrego y Dr. David Benavides
Lenguajes y Sistemas Informáticos
Universidad de Sevilla,
Sevilla, junio de 2022

Yo, Mariuxi Geovanna Vinueza Morales, con número de DNI 0917189664,

DECLARO

Ser la autora del trabajo que se presenta en la memoria de esta tesis doctoral que tiene por título:

Evidencias empíricas en el proceso de enseñanza de la programación. Una aproximación usando ALICE en el contexto universitario

Lo cual firmo en Sevilla, junio de 2022.

Fdo. Mariuxi Geovanna Vinueza Morales

A mi familia
Remigio, Isabel, Pily y Mary
En especial a mis hijos Giovanna y Javier

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Al finalizar este arduo y esforzado trabajo de investigación como es una tesis doctoral, quiero expresar de manera especial y sincero agradecimiento a mis tutores Diana Borrego y David Benavides por aceptar dirigir este trabajo, bajo su guía y dirección pero por sobre todo su apoyo y confianza. A mis amigos José A. Galindo y Bedilia Estrada por su valiosa ayuda y colaboración a lo largo del desarrollo de ésta tesis.

Expreso también un sincero agradecimiento a la familia UNEMI, de manera especial a Fabricio y Jesennia, por la oportunidad brindada para aprender y progresar en el ámbito profesional de la investigación.

Sin duda terminar con este trabajo de investigación, no ha sido un tarea fácil, es parte de un largo período de aprendizaje y resultado de esfuerzos y sacrificios que al final de la meta me llena de orgullo y satisfacción.

Gracias a mi familia, a mis padres, mis hermanas y en especial a mis hijos Giovanna y Javier, quienes son mi apoyo vital y en todo momento han demostrado comprensión por las innumerables horas invertidas en mi trabajo doctoral.

Finalmente gracias a la vida por este nuevo triunfo.

Resumen

Aprender a programar es una de las habilidades fundamentales para los estudiantes relacionados con el área de la informática. La comprensión de los conceptos básicos relativos a esta materia son fundamentales en la evolución del aprendizaje. Sin embargo, se han identificado varios problemas en el aprendizaje de la programación como por ejemplo la dificultad en la comprensión y resolución de problemas, y estudiantes poco motivados en el desarrollo de competencias de programación.

Para algunos autores es necesario que se analice el problema en el aprendizaje de la programación, debido a un alto porcentaje de abandono por parte de los estudiantes que cursan las asignaturas de programación, Otras investigaciones realizadas sustentan que el uso de un lenguaje de programación tradicional ha presentado deficiencias en el proceso y por consiguiente influye en la falta de capacidad de habilidades de resolución de problemas.

Comprender los conceptos computacionales a través de herramientas como un lenguaje de programación educativo, permite a los alumnos novatos de los primeros niveles de las carreras de informática, adquirir conceptos de programación y desarrollar aplicaciones con facilidad. Incorporar una herramienta como parte del diseño pedagógico del proceso de aprendizaje de la programación puede crear algunas ventajas en la actitud del estudiante como incrementar su desempeño, mantenerse motivado y comprometido en aprender conceptos de programación.

En este ámbito, surge la necesidad de conocer los lenguajes de programación aplicados en un contexto educativo. Por ello, en la presente investigación se realizó un trabajo de revisión de literatura para obtener evidencias empíricas de la aplicación de herramientas de programación educativas en el proceso de aprendizaje de la programación. Entre los resultados de la revisión sistemática se identificó los métodos utilizados y la calidad del método, el contexto educativo de los trabajos analizados, los objetivos pedagógicos descritos en la literatura y las alternativas para evaluar la eficacia del aprendizaje luego de la aplicación del lenguaje de programación educativo.

Adicional a esto, se realizó una intervención empírica donde se hizo un primer trabajo piloto por medio de un experimento para identificar la reacción

de los estudiantes en la aplicación de un lenguaje de programación educativo. De los trabajos identificados en la literatura, se detectó que no había suficiente evidencia empírica sobre el software ALICE en el contexto universitario. Por este motivo, se consideró aplicar ALICE en el experimento con estudiantes novatos en nivel universitario. Los resultados presentan que ALICE tuvo un efecto positivo en la formación de competencias de los estudiantes de educación superior, además se evidencia que hay un incremento en la asimilación de conceptos de programación y una reducción de fracasos en el aprendizaje.

Por otra parte, la mayoría de los trabajos descritos en la literatura se realizaron en sociedades WEIRD, occidentales, educadas, industrializadas, ricas y democráticas (*Western, Educated, Industrialized, Rich and Democratic Societies*), lo que representa una realidad distinta en países que no pertenecen a la comunidad (WEIRD) como es el caso de Ecuador, donde se realizó la investigación de la presente tesis.

Finalmente, con el presente trabajo se determinó que el uso de los lenguajes de programación educativos son una alternativa para incrementar el pensamiento computacional en los estudiantes. Además, pueden ser implementados en el programa de estudio en las asignaturas de programación. Sin embargo, hace falta más evidencia empírica en un contexto universitario, siendo esto una oportunidad de investigación para un trabajo futuro.

Abstract

Learning to program is one of the fundamental skills for students related to the area of computer science. The understanding of the basic concepts related to this subject are fundamental in the evolution of learning. However, several problems have been identified in learning programming, such as difficulty in understanding and solving problems, and students who are poorly motivated to develop programming skills.

For some authors it is necessary to analyze the problem in learning programming, due to a high percentage of students who abandon the study of programming subjects. Other researches sustain that the use of a traditional programming language has presented deficiencies in the process and therefore influences the lack of problem-solving skills.

Understanding computational concepts through tools such as an educational programming language allows novice students in the first levels of computer science degrees to acquire programming concepts and develop applications with ease. Incorporating a tool as part of the pedagogical design of the programming learning process can create some advantages in the attitude of the students such as increasing their performance, staying motivated and engaged in learning programming concepts.

In this area, the need arises to know the programming languages applied in an educational context. Therefore, in the present research, a literature review was carried out to obtain empirical evidence of the application of educational programming tools in the learning process of programming. Among the results of the systematic review were identified the methods used and the quality of the method, the educational context of the analyzed works, the pedagogical objectives described in the literature and the alternatives to evaluate the effectiveness of learning after the application of the educational programming language.

In addition to this, an empirical intervention was carried out where a first pilot work was done by means of an experiment to identify the reaction of students in the application of an educational programming language. From the works identified in the literature, it was detected that there was not enough empirical evidence on ALICE software in the university context. For this reason, it

was considered to apply ALICE in the experiment with novice students at the university level. The results show that ALICE had a positive effect on the adquisition of skills of higher education students, and it is also evident that there is an increase in the assimilation of programming concepts and a reduction of learning failures.

On the other hand, most of the works described in the literature were carried out in WEIRD (Western, Educated, Industrialized, Rich and Democratic Societies), which represents a different reality in countries that do not belong to the WEIRD community, as is the case of Ecuador, where the research of this thesis was carried out.

Finally, with the present work it was determined that the use of educational programming languages is an alternative to increase computational thinking in students. In addition, they can be implemented in the curriculum in programming subjects. However, more empirical evidence is needed in a university context, being this a research opportunity for future work.

Parte I

Preliminares

Capítulo 1

Introducción

Todos en este mundo deberían aprender a programar... porque te enseña a pensar.
Steve Jobs

En esta tesis, se reporta el trabajo realizado sobre el uso de Lenguajes de Programación Educativos (LPEs) para el aprendizaje de la programación, además se realizó la evaluación de una herramienta de aprendizaje mediante la aplicación en el aula con estudiantes de nivel superior. En la Sección §1.1 se establece el contexto general de la investigación; más adelante en la Sección §1.2 se presenta un breve resumen de las principales contribuciones. Luego, en la Sección §1.3 se presentan una breve descripción de la organización del documento.

1.1. Contexto de la investigación

El desarrollo tecnológico, especialmente de las Tecnologías de la información y Comunicación (TIC), presenta un impacto significativo en la economía y otros aspectos de la vida humana en las últimas décadas; por lo que es imposible imaginar el funcionamiento efectivo de un individuo, una economía y toda una sociedad sin el uso de ellas [85]. Las TIC se han convertido en elementos típicos en todos los ámbitos de la vida; durante los últimos veinte años, su utilización ha cambiado en general las prácticas y estrategias de casi todo tipo de empresas y administraciones [146]; han contribuido al desarrollo de nuestra sociedad, transformando drásticamente la forma en que vivimos [30].

Ese impacto que han tenido las TIC en cada uno de los espacios de la sociedad ha generado, a su vez, la necesidad de competencias y habilidades en su manejo, haciendo que sean las instituciones de educación superior las llamadas a asumir este reto. Así, la inclusión de las TIC en los contenidos analíticos de las instituciones educativas podría causar un efecto favorable en las prácticas de enseñanza y aprendizaje, mejorando la calidad en la educación con el uso de la tecnología y métodos modernos de enseñanza, enfocar una educación con mejores resultados de aprendizaje y reformas en los contenidos del sistema educativo [53].

1.1.1. Pensamiento computacional

Dentro de las competencias y habilidades alrededor de las TIC, se destaca el pensamiento computacional como un motor de esas necesidades que están demandando en la sociedad actual. El pensamiento computacional, el enfoque para resolver problemas como un informático o ingeniero de software, se considera una competencia crítica para los trabajadores del conocimiento del siglo XXI [178]; una habilidad que los individuos deben adquirir y utilizar para resolver eficientemente los problemas que se encuentran en la vida [71]; en los últimos años, ha sido descrito y reconocido también como una habilidad esencial para los ciudadanos [37].

Contribuir para que los planes de estudio incluyan contenidos de ciencias de la computación y pensamiento computacional es un avance que enfatiza la práctica de auténticas aptitudes del pensamiento computacional, considerando principalmente el contexto de los entorno de programación basados en bloques [66]. La informática comprende las habilidades y prácticas tanto de la ciencia de la computación como del pensamiento computacional. Mientras, que las ciencias de la computación es una disciplina académica individual, el pensamiento computacional es un enfoque de resolución de problemas que se integra en todas las actividades.

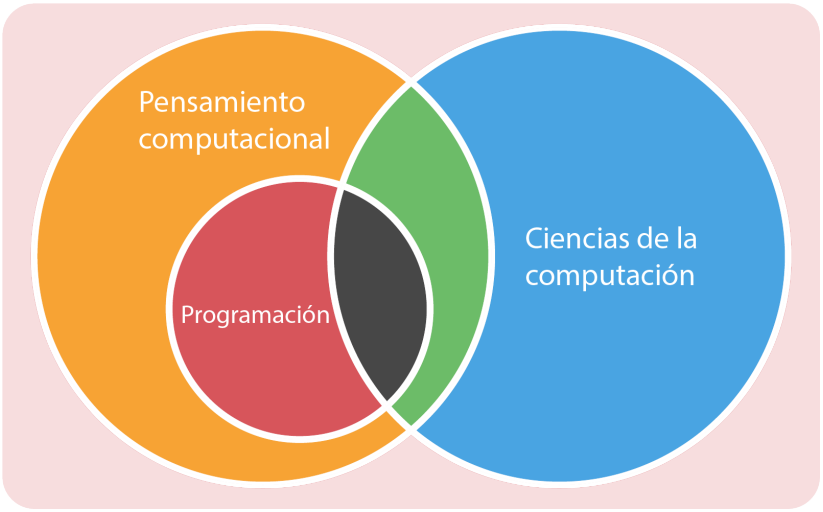

Figura 1.1: Relación entre programación, ciencias de la computación y pensamiento computacional [11]

El pensamiento computacional involucra conceptos y razonamientos fundamentales, destilados de la informática y otras ciencias computacionales, que se convierten en poderosas herramientas mentales generales para resolver problemas, aumentar la eficiencia, reducir la complejidad, diseñar procedimientos o interactuar con humanos y máquinas [166]. Dentro de las estrategias para desarrollar competencias en el pensamiento computacional se destaca el aprendizaje de la programación, donde, dadas las diversas dificultades encontradas en los nuevos profesionales de ciencias de la computación, se han presentado una variedad de herramientas para su aprendizaje efectivo.

En la Figura §1.1 presenta la relación entre la ciencia de la computación, el pensamiento computacional y la programación. El pensamiento computacional es una habilidad cognitiva que facilita la resolución de problemas y que se integra en todas las actividades, en tanto la programación consiste en desarrollar un conjunto de instrucciones que serán ejecutadas en una computadora para brindar de manera apropiada soluciones a los problemas.

1.1.2. Lenguajes de programación educativos

La programación se ha hecho mas popular en el entorno educativo y está cada vez está más presente en las aulas de clases. Su aplicación aporta a los alumnos una mayor capacidad lógica y analítica, y su aprendizaje promueve el desarrollo de habilidades como el pensamiento computacional, creatividad, innovación y las destrezas para crear herramientas y resolver problemas a través de

la tecnología.

Se conoce que en los últimos años, se han propuesto diversas herramientas y recursos para incorporar conceptos introductorios de programación y desarrollar el pensamiento computacional en las aulas de clases. Entre las herramientas están los Lenguajes de Programación Educativos (LPEs) como ALICE, Scratch, AppLab, App Inventor entre otros; en el Capítulo §3 se mencionan algunos de ellos.

Para establecer algunas de los mecanismos utilizados en el proceso de aprendizaje de la programación se realizó un trabajo de Revisión Sistemática de la Literatura (RSL), de aquellos trabajos que presentaban evidencia empírica de los resultados obtenidos, de los cuales se identificó: los lenguajes de programación utilizados, los objetivos pedagógicos alcanzados y los métodos empíricos utilizados en cada estudio, en el Capítulo §6 se presenta información sobre la revisión de literatura realizada.

Los lenguajes de programación identificados en la RSL, fueron clasificaron en cuatro categorías: 1) lenguajes basados en bloques, 2) lenguajes de programación textuales; 3) juegos para la enseñanza del pensamiento computacional; y, 4) otras herramientas educativas.

Con la base de las categorías antes mencionadas, para el presente trabajo se consideró los *lenguajes de programación basados en bloques (VPL)*, la finalidad es seleccionar uno de los lenguajes educativos para aplicarlo en un experimento y determinar la efectividad del mismo en el proceso de aprendizaje de la programación, con estudiantes en un nivel universitario. De los lenguajes de programación basados en bloques se destacan Scratch, AppInventor y ALICE. En el Capítulo §7 se detalla el experimento realizado.

En el experimento que se desarrolló para el presente trabajo de investigación, se seleccionó el lenguaje de programación ALICE. Algunas investigaciones [5, 6, 40, 51] consideran ALICE como una herramienta que ofrece métodos innovadores para la enseñanza de la programación. ALICE está escrito en Java y tiene una filosofía orientada a objetos. En [42, 161] se realiza una revisión de literatura sobre la utilidad de ALICE. Estas investigaciones presentan información de que el uso de ALICE tiene incidencia en la motivación del aprendizaje de la programación, la resolución de problemas y el pensamiento algorítmico, con posibles efectos positivos, de que ALICE como estrategia de enseñanza presentaría resultados mas eficaces que la aplicación de un lenguaje de programación convencional.

1.1.3. Enseñanza de la programación con ALICE

ALICE es un entorno de programación, propuesto por la Universidad Carnegie Mellon (CMU)[†1], usado para introducir la tecnología 3D con animaciones gráficas, juegos interactivos y vídeos compartidos en la web, con fundamentos en el proceso de aprendizaje de la programación orientada a objetos (POO).

ALICE es un software libre y está escrito en Java, se puede ejecutar en Windows, Linux o MacOS. Hay dos versiones disponibles de ALICE para su descarga: Alice 2[†2], maneja un entorno menos complejo y está dirigido a estudiantes de secundaria y preparatoria, y Alice 3, que puede ser utilizado por estudiantes mas avanzados debido a sus características para ayudar en una transición completa al lenguaje de programación Java.

El entorno ALICE permite acceder al potencial de la programación en un entorno visualmente atractivo, sin requerir una comprensión previa de la sintaxis y los conceptos de programación orientada a objetos [168]. ALICE posee una interfaz interactiva donde se arrastran y sueltan cuadros y gráficos para crear un programa, las instrucciones que se generan son sentencias estándares de un lenguaje de programación orientado al desarrollo, como Java, C++ y C# [65].

ALICE tiene un enfoque orientado a objetos, hace énfasis en conceptos de programación como objetos, clases, herencia, condiciones, bucles, variables, matrices, eventos y recursividad. Los usuarios de ALICE formulan programas utilizando segmentos de código de arrastrar y soltar en lugar de escribir y preocuparse por la sintaxis del lenguaje, pudiendo también escribir simples scripts y controlar la apariencia y comportamiento de los objetos. ALICE se puede utilizar como un lenguaje de programación para el programador novato ya que pueden visualizar inmediatamente sus programas animados. Además es fácil detectar los errores en los programas y corregirlos de una manera sencilla.

En la Tabla §1.1 se presentan los trabajos de [4, 14, 52, 66, 107], en los cuales se utilizó ALICE en el proceso de aprendizaje como una herramienta para enseñar a programar a los estudiantes de distintos niveles educativos. Se reporta que, mediante talleres y pruebas piloto, evaluaron su efectividad como una herramienta educativa. Los participantes consideraron que aprendieron mucho y tuvieron una experiencia positiva con el software ALICE.

Por otra parte, Al-Tahat y otros [7], Biju [22], Dwarika y De Villiers [51], Johnsgard y McDonald [79] evaluaron el efecto de ALICE en la actitud de los estudiantes hacia las clases recibidas y su rendimiento en el curso introductorio

[†1] https://www.alice.org/
[†2] https://www.alice.org/get-alice/alice-2/

Citas	Año	País	Nivel
Assiter y Wiseman [14]	2016	USA	Secundaria (K-12)
Al-Sabbagh y otros [4]	2017	Qatar	Secundaria (8vo grado, K-12)
Al-Tahat y otros [7]	2016	Jordania	Universidad, 1er. año
Biju [22]	2013	Emiratos Árabes Unidos	Universidad, 1er. año
Dwarika y De Villiers [51]	2015	Sudáfrica	Universidad, 2do. año
Edwards y otros [52]	2007	USA	Universidad, 1er. año
Grover y otros [66]	2017	USA	Secundaria (K-12)
Johnsgard y McDonald [79]	2008	USA	Universidad
Morales Diaz y otros [107]	2015	México	Universidad

Tabla 1.1: Listado de trabajos desarrollados con ALICE [161]

de programación, como resultados de estos estudios se consideró que ALICE tuvo un impacto positivo y fue una experiencia valiosa en los participantes hacia el aprendizaje de la programación. En los estudios mencionados, trabajaron con dos grupos, un grupo experimental usando la herramienta ALICE y otro grupo de control sin ALICE.

En cuanto a la región, la mayoría de los trabajos presentados en la Tabla §1.1, se han realizado en Estados Unidos, con escasos estudios en Europa y Latinoamérica. En cuanto a los niveles educativos donde se realizaron los trabajos, están el nivel secundario (K-12) y la universidad (1er. y 2do. año). Sin embargo, es necesario que se incrementen las iniciativas de estudios en países non-WEIRD para establecer las necesidades reales y vacíos que se tiene en el proceso de aprendizaje de la programación.

1.2. Contribuciones

En esta sección se presenta un resumen de las principales contribuciones del presente trabajo, que han sido publicados en diferentes revistas y conferencias. Entre las principales contribuciones de este trabajo doctoral, se destacan:

1. Las revisiones sistemáticas de literatura, tanto de los entornos de aprendizaje de programación, como el caso particular de dos lenguajes de programación educativos; los cuales están enmarcadas dentro de la Ingeniería de Software basada en evidencia.

   En este fase se realizaron varias revisiones sistemáticas de literatura de la aplicación de los programas educativos en el proceso de aprendizaje en ingeniería del software, y sobre la utilidad del software ALICE y App Inventor, como herramientas en el proceso de aprendizaje de programación. Los resultados sirvieron de soporte a este trabajo doctoral y como

motivación para su aplicación en la siguiente fase.

2. Las experiencias de aplicación de plataformas de aprendizaje; en particular un lenguaje de programación educativo, en un curso introductorio a la programación en la Universidad Estatal de Milagro, Ecuador.

   En esta fase cuasiexperimental, se seleccionó el lenguaje de programación ALICE, se diseñaron dos grupos de sujetos: control, con el enfoque tradicional; y experimental, con el entorno ALICE. Los resultados evidencian que el grupo experimental presentó resultados mas destacados, ya que en promedio de notas obtuvieron 80, mientras que el grupo de control fue de 65 puntos; esto sugiere la efectividad del entorno ALICE y favorece el proceso de aprendizaje de programación en estudiantes de un nivel universitario del área de informática.

A continuación se presentan las publicaciones en orden cronológico:

[2019]

El uso de software ALICE como herramienta para el aprendizaje de programación: una revisión de literatura, *International Multi-Conference for Engineering, Education Caribbean Conference for Engineering and Technology-LACCEI*. http://dx.doi.org/10.18687/LACCEI2019.1.1.161

En esta publicación se analiza el uso de ALICE para determinar si es una herramienta apropiada para aplicar en el proceso de aprendizaje de programación, dada su utilidad para desarrollar habilidades de pensamiento algorítmico y de resolución de problemas.

Por otra parte, se presentan los resultados de la revisión sistemática de la literatura, de trabajos que han utilizado ALICE como lenguaje de programación en un escenario educativo. Aplicando la cadena de búsqueda y algunos criterios se seleccionaron 24 artículos para ser analizados, los mismos que fueron extraídos de las bases de datos del área de ciencias de la computación como ACM, IEEE, entre otras.

Entre los resultados se destaca las evaluaciones de efectividad del uso de ALICE como herramienta para la introducción a la programación, la cantidad de trabajos en inglés y la débil validez de los resultados experimentales.

[2020]

Enseñanza de programación mediante MIT App Inventor: una revisión de literatura. *18th LACCEI International Multi-Conference for Engineering, Education and Technology: Engineering, Integration and Alliances for a Sustainable Development Hemisferic Cooperation for Competitiveness and Prosperity on a Knowledge-Based Economy*. http://dx.doi.org/10.18687/LACCEI2020.1.1.49

En esta publicación se analiza a MIT App Inventor, una herramienta que se basa en la programación visual por bloques, su uso está orientado preferentemente para que los programadores novatos aprendan conceptos de programación. Su enfoque es la creación de aplicaciones para móviles. Se realiza una revisión de literatura utilizando motores de búsqueda especializados, en los resultados se seleccionaron y analizaron 35 estudios, que se aplicaron en distintos países sobre el proceso de enseñanza-aprendizaje de programación.

Los resultados presentan una aceptación positiva del App Inventor para introducir conceptos de programación, incrementar la motivación y el rendimiento de los estudiantes sin distinguir el nivel educativo.

[2020]

Emprirical Evidence of the Usage of Programming Languages in the Educational Process. *IEEE Transactions on Education*, 64 (3), 213-222. http://dx.doi.org/10.1109/TE.2020.3030588

En esta publicación se presenta una RSL, sobre la evidencia empírica del uso de lenguajes de programación con fines de aprendizaje. La revisión analiza diferentes métodos y herramientas en diferentes niveles educativos y con diferentes objetivos. Siguiendo un protocolo formal, se realizaron búsquedas automatizadas de estudios primarios desde 2007 hasta 2018. Se identificaron un total de 62 estudios, de los cuales se seleccionaron y analizaron 29 por incluir algún tipo de evidencia empírica.

Después de realizar la evaluación, los resultados respaldan la necesidad de mejores enfoques con evidencia empírica al informar investigaciones sobre el uso de Lenguaje de Programación Educativo (LPE). Se identifican algunas oportunidades de investigación relacionadas con los lenguajes de programación utilizados, las áreas o etapas de su aplicación, o la necesidad de tener más evidencia en general y más estudios en contexto non WEIRD (occidentales, educados, industrializados, ricos y democráticos), por ejemplo países de Sudamérica como Ecuador, Perú, Colombia, entre otros. En el Capítulo §6 se detalla información correspondiente a la revisión de literatura realizada.

1.2.1. Método de investigación

Los métodos de investigación utilizados en este trabajo doctoral son dos, básicamente: la revisión sistemática de la literatura y el experimento para la evaluación de un lenguaje de programación educativo.

La RSL se ha adoptado dentro de la ingeniería de software durante más de una década para proporcionar resúmenes significativos de evidencia sobre varios

temas [176]. Es una herramienta metodológica que juega un papel importante en la ingeniería de software basada en evidencia [16].

Una RSL consiste en identificar, analizar y reportar información relevante y útil sobre trabajos de investigación disponibles con respecto a un tema que se investiga, en el área de interés [87]; siguiendo una metodología rigurosa y auditable para minimizar los sesgos y garantizar la confiabilidad de los resultados [122].

En el presente trabajo se realizó una RSL sobre la evidencia empírica referente al uso de los LPEs y su incidencia en el aprendizaje de la programación en distintos niveles educativos y con diferentes objetivos de investigación. Se siguió un protocolo formal, con búsquedas automáticas de trabajos desde 2007 hasta 2018. Para ello, se identificaron un total de 62 fuentes, de las cuales se seleccionaron y analizaron 29, que incluyen algún tipo de evidencia empírica.

Siguiendo las directrices de [88], en la RSL se desarrollaron las siguientes fases,

(1) Planificar la RSL. Esta fase incluye las actividades relacionadas con la organización y la validación del protocolo.

(2) Ejecutar la RSL. Esta fase incluye la ejecución del protocolo de revisión y el seguimiento de su progreso.

(3) Documentar la RSL. Esta fase incluye las actividades, como la documentación o el informe del estudio de forma adecuada para los destinatarios.

Posteriormente, considerando los resultados de la RSL, se realizó una evaluación de la efectividad de uno de los LPEs en el proceso de aprendizaje de la programación en estudiantes universitarios. Se seleccionó el experimento como metodología para realizar el estudio empírico, siguiendo para la realización de este trabajo las directrices de Wohlin y otros [177], en el cual se trabajó bajo condiciones controladas con dos grupos de estudiantes (experimental y de control) para así poder establecer diferencias significativas que nos permitan determinar el éxito del aprendizaje.

El experimento se desarrolló en cuatro fases, que se detallan a continuación:

(1) Definir alcance: como primera fase se identifica la necesidad de llevar a cabo el experimento, y se aborda la determinación de su propósito y objetivos;

(2) Planificar: en esta fase se diseña y planifica el experimento, preparando las herramientas necesarias para poder desarrollarlo. Concretamente, se determina en detalle el contexto del experimento, se establecen las hipótesis de partida, y se definen los sujetos objeto del estudio así como el material necesario para llevarlo a cabo;

(3) Realizar operación: en esta fase se ejecuta el experimento de acuerdo al plan y diseño establecidos, y se recaban los datos para su procesamiento posterior;

(4) Analizar e Interpretar: se analizan los datos obtenidos durante y después del experimento, proporcionando una visualización de los mismos, y se analizan los resultados para obtener una conclusión final.

.

1.3. Estructura de la tesis

Este documento está organizado de la siguiente manera:

Parte I. Preliminares: En la primera parte de la tesis, se presenta el Capítulo §1 Introducción, que incluye una visión general del contexto de la investigación, un enunciado de las contribuciones y el método de investigación, las principales publicaciones y la estructura general.

Parte II. Antecedentes: En la segunda parte de la tesis, se incluyen los temas de pensamiento computacional, herramientas para el aprendizaje de la programación, y acerca del entorno de programación ALICE.

En el Capítulo §2 sobre pensamiento computacional, se incluye conceptualización, competencias e implicaciones en el contexto educativo. A continuación, en el Capítulo §3 sobre herramientas para el aprendizaje de la programación, se aborda el uso de herramientas informáticas en el contexto educativo, experiencias en el contexto de la enseñanza de programación, los entornos de programación visual basada en bloques; y finalmente, en el capítulo §4 sobre el entorno de programación ALICE, se describe ALICE como lenguaje orientado a objetos, y se profundiza acerca de las fortalezas de ALICE y las experiencias con su uso.

Parte III. Contribuciones: En esta parte, se incluye en el Capítulo §5 la motivación de la investigación doctoral; a continuación, en el Capítulo §6, se aborda la RSL sobre el uso de los lenguajes de programación en el proceso educativo; y por último, en el Capítulo §7, se exponen los resultados del experimento del uso de ALICE como herramienta para el aprendizaje de la programación en nivel universitario.

Parte IV. Conclusiones finales: Se incluye las conclusiones y trabajos futuros sobre nuevos problemas que procedan de este trabajo de tesis.

Parte V. Apéndice: En el Anexo A se encuentra las cadenas de búsqueda aplicadas a las bases de datos.

Parte II

Antecedentes

Capítulo 2

Pensamiento computacional

La única habilidad competitiva a largo plazo es la habilidad para aprender.
Seymour Papert, 1997

El pensamiento computacional es un proceso que influye en cada una de las competencias en las cuales debe formarse un profesional de ciencias de la computación, y sobre cualquier individuo que quiera interactuar en este ecosistema tecnológico del siglo XXI. Aunque se puede definir el pensamiento computacional como un proceso de resolución de problemas con base en recursos de ciencias de la computación, su carácter multidisciplinario, complejo y transversal, hace que sea necesaria una revisión de las diferentes iniciativas para su conceptualización, estructuración y operacionalización, así como las implicaciones en el contexto educativo; este es, precisamente, el propósito de éste capítulo.

En la Sección §2.1 se especifica una introducción al pensamiento computacional; más adelante en la Sección §2.2 se conceptualiza el pensamiento computacional. Luego, en la Sección §2.3 se hace un análisis de las competencias del pensamiento computacional; para luego en la Sección §2.4 presentar las implicaciones del pensamiento computacional, en el contexto educativo. Finalmente, resumimos este capítulo en la Sección §2.5.

2.1. Introducción

El contexto moderno demanda implementar novedosos métodos de enseñanza en las aulas de clases, teniendo en cuenta los avances tecnológicos e impulsando estrategias para alcanzar competencias digitales, que en general faciliten a los estudiantes desenvolverse en una comunidad tecnológica [63]. Aplicar las TIC en todos los campos requiere la capacidad de pensamiento computacional, que ahora es una tendencia; ésta capacidad fundamental debe ser propiedad de todos los estudiantes en la era digital [69]. En la última década, éste tema ha ganado cada vez más atención por parte de investigadores y profesionales en el campo de la educación [24].

El pensamiento computacional es el enfoque para resolver problemas como un informático o ingeniero de software. Se estima que sea una competencia crítica para los trabajadores del conocimiento del siglo XXI [178]; se considera una habilidad que las personas comunes deben adquirir y utilizar para dar solución eficientemente a los problemas que se encuentren en el día a día [71]; en los últimos años, ha sido descrito y reconocido también como una habilidad esencial para los ciudadanos [36].

En éste sentido, hay muchas instituciones educativas que han incluido de manera obligatoria en sus planes de estudio el tema del pensamiento computacional [2]; implementando un enfoque moderno, novedoso y esencial en los planes de estudio de todos los niveles educativos [139]; y, consolidando así, que a nivel internacional se establece el tema de introducir el pensamiento computacional, así como la programación y la robótica en las instituciones educativas [159].

Así, en respuesta a los avances inminentes hay un enfoque renovado en las instituciones educativas para enseñar a los estudiantes que colaboren activa y productivamente en una sociedad cada vez más controlada por la tecnología digital [12]. Aunque es recientemente cuando el pensamiento computacional ha recibido un reconocimiento significativo para la comunidad educativa, su noción no es nueva, estando presente en el discurso académico bajo diferentes formas durante décadas [32]; sin embargo, es en los últimos años cuando ha desarrollado un fuerte avance teórico [60].

El pensamiento computacional ha sido un sello distintivo de la informática desde la década de los 1950's Denning [48]. En cuanto a sus antecedentes recientes, los orígenes en el contexto de ciencias de la computación se remontan a los trabajos de Papert (1980, 1991), quien relacionó la práctica de la informática con el acto de pensar, tema central de su trabajo con el lenguaje de programación LOGO Yadav y otros [182].

Aunque algunos autores como Aranda y Ferguson [12], Segredo y otros [139],

y Ulrich Hoppe y Werneburg [157], entre otros, coinciden en señalar que las raíces del pensamiento computacional reciente se remontan a los trabajo de Papert, la mayoría se refiere a la propuesta de Wing [173], quien popularizó el tema [182]; aunque no está claro si el planteamiento de Papert se refería a la misma construcción planteada por Wing Voogt y otros [163].

Actualmente, se reconoce la importancia del pensamiento computacional, que se hace necesario su conceptualización en el marco del contexto de éste trabajo doctoral; por lo que, en la sección siguiente, se presentan algunos intentos por establecer una definición de pensamiento computacional, así como, su estructura y características.

2.2. Pensamiento computacional: conceptualización

La propuesta del concepto de pensamiento computacional ha iniciado una acalorada discusión alrededor del mundo Wang y Feng Wang [165]. Las primeras conceptualizaciones del pensamiento computacional surgieron en un momento donde habían muchas dudas sobre el naciente campo de las ciencias informáticas, y especialmente sobre su diferencia con otros campos más establecidos Tedre y Denning [152].

A pesar de este renovado interés en el pensamiento computacional, los profesionales en el contexto educativo se enfrentan a una serie de problemas y desafíos cuando intentan integrar el pensamiento computacional en el currículo Voogt y otros [163]. A falta de una definición única de este campo, surgen varias conceptualizaciones de la literatura para tratar de entender sobre pensamiento computacional; éstos conceptos incluyen temas como: abstracción, pensamiento algorítmico, automatización, descomposición, depuración y generalización Bocconi y otros [24].

Mientras que la demanda de incluir el pensamiento computacional en los planes de estudio ha incrementado, los estudiantes que cursan las asignaturas de programación en un nivel universitario deberían aprender como desarrollar la lógica mediante actividades de aprendizaje utilizando lenguajes de programación visual basados en bloques. Esta sería una experiencia que les permitiría probar otras formas para adquirir conceptos básicos de programación.

Por otra parte, en Harimurti y otros [68] se establece que el pensamiento computacional es una habilidad fundamental para todos los individuos y también se mencionan los elementos del pensamiento computacional 1) descomposición; descomponer un problema completo en tareas sencillas, 2) reconocimiento de patrones: identificar semejanzas y deferencias entre patrones, 3) abstracción: reconocer los principios general de los patrones; y, 4) algoritmo:

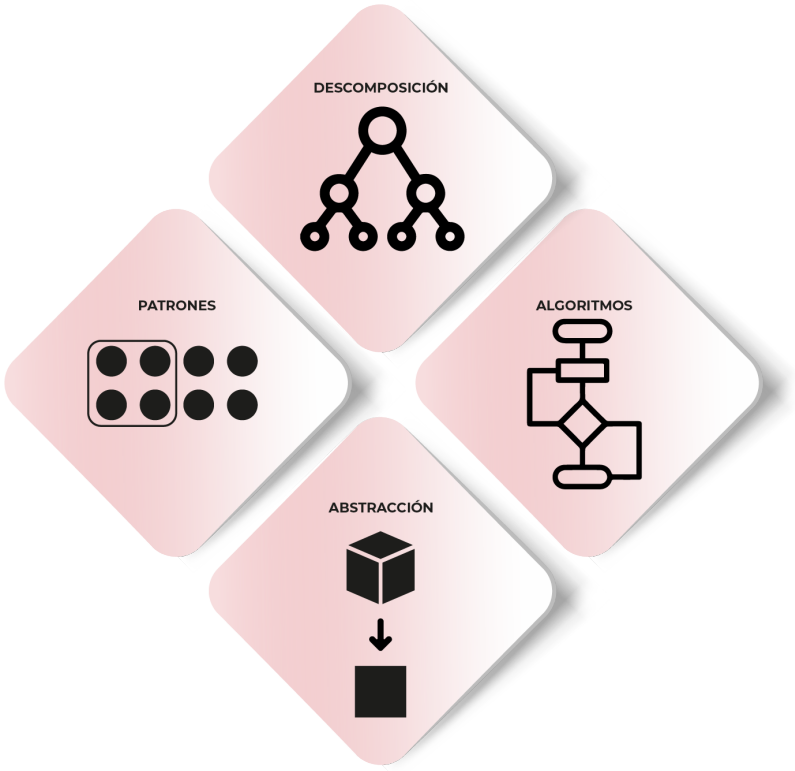

Figura 2.1: Elementos del pensamiento computacional [68]

desarrollar la solución paso a paso para que pueda reutilizarse. En la Figura §2.1 se presentan los elementos antes mencionados.

El pensamiento computacional es un proceso que comprende la identificación y resolución de problemas, el diseño de sistemas y la comprensión de la conducta de los individuos por medio de conceptos básicos de informática, con la ayuda de herramientas tecnológicas Wing [173]. Básicamente, el pensamiento computacional es un proceso iterativo y cíclico de razonamiento deductivo e inductivo, tal como lo emplean los científicos en la investigación y los estudiantes en el aprendizaje Yasar [183]; y, de acuerdo con Selby y Woollard [143], parece haber un consenso de que una definición de pensamiento computacional debe incluir la idea de un proceso de pensamiento, el concepto de abstracción y el concepto de descomposición.

Como lo planteó Wing [173], en su etapa de conceptualización inicial, el pensamiento computacional se fundamenta en el alcance de los procesos informáticos, ejecutado por una máquina o realizado por un humano; y, aunque, el término es relativamente nuevo, el proceso implícito por Wing se puede reconocer como una versión computacional mejorada del método científico Re-

penning y otros [125]; y, que complementa el pensamiento crítico como una forma de razonar para resolver problemas, tomas decisiones e interactuar con el entorno [92].

Para Bocconi y otros [24], el pensamiento computacional es el término en uso actual para referirse a las ideas y conceptos clave de la computación y la informática; Buitrago y otros [28], indican que representa una forma de abordar situaciones cotidianas y resolver problemas utilizando conceptos que son fundamentales para la informática. Mientras que para Denning [48], representan los hábitos de la mente desarrollados a partir del diseño de programas, paquetes de software y cálculos realizados por las máquinas.

El pensamiento computacional es una proceso de pensamiento que ofrece soluciones a los problemas por medio del desarrollo de destrezas como la descomposición, la abstracción, la evaluación, la generalización y el diseño de algoritmos Rojas López y García-Peñalvo [130]. Por su parte Nardelli [111], lo interpreta como la capacidad de pensar como un científico informático y ser capaz de aplicar esta competencia a todos los campos del esfuerzo humano; mientras que Haseski y otros [71], afirman que es una habilidad que desarrollan todos los individuos en general y permite tomar decisiones activas y sistemáticas utilizando las TIC y los enfoques de colaboración en situaciones de la vida real, y así alcanzar las decisiones mas ideales y éticas y contribuir a su entorno, al autodescubrimiento y a la satisfacción de los individuos con autoestima.

Por otra parte, en Voogt y otros [163] consideran que el pensamiento computacional está basado en los principios de la informática, sin embargo, no son lo mismo; la informática es considerada una ciencia que estudia las computadoras (hardware), sistemas computacionales (software) y redes, en tanto que el pensamiento computacional contempla procesos del pensamiento que permite la resolución de problemas complejos y la descomposición de este proceso en una diversidad de problemas.

El pensamiento computacional conlleva la formulación del problemas, descomponerlo y ofrecer una solución por medio del uso de algoritmos; abstraer y automatizar el enfoque de resolución de problemas [182]. Para *The British Royal Society [154],* el pensamiento computacional consiste en la capacidad de identificar características del área de la computación, y determinar que técnicas y herramientas informáticas se deben aplicar para asimilar información concerniente a sistemas naturales y artificiales.

Por su parte, Selby y Woollard [143] lo definen como una actividad basada en el cerebro que permite resolver problemas, comprender las situaciones y expresar mejor los valores mediante la aplicación y evaluación en la producción de una automatización implementable por un dispositivo informático digi-

tal o humano. Para Selby y Woollard [142], el pensamiento computacional se centra en la resolución de problemas e incorpora procesos de desarrollo del pensamiento utilizando abstracción, descomposición, algoritmos, evaluación y generalizaciones; estos conceptos están asociados con destrezas del modelado, simulación y visualización, pero no está definida por ellas.

Lee y otros [93] sostienen que, el pensamiento computacional implica definir, comprender y resolver problemas, razonar en múltiples niveles de abstracción, comprender y aplicar la automatización, y analizar lo apropiado de las abstracciones realizadas; mientras que Aho [3], considera que, el pensamiento computacional es el proceso del pensamiento implicado en la identificación de problemas, pudiendo representar las soluciones en algoritmos computacionales. Adicionalmente, una parte importante de este proceso es encontrar modelos apropiados de cómputo con los cuales formular el problema y derivar sus soluciones.

El pensamiento computacional es considerado parte del pensamiento analítico el cual junto con el pensamiento matemático coinciden con la forma de resolver problemas; por otra parte, con el pensamiento de la ingeniería establecen la manera en que se diseña y evalúa un sistema complejo del mundo real con las limitaciones respectivas; finalmente con el pensamiento científico comparte criterios en cuanto a la comprensión de la computarización, la mente, la inteligencia y el comportamiento humano [174]. En otra perspectiva, Lee y otros [93], plantean que el pensamiento computacional "comparte elementos con otros tipos de pensamiento matemático". El pensamiento computacional se distancia de la competencia/alfabetización digital, ya que se centra en los procesos y formas de resolver los problemas y en proponer soluciones computables [24].

Segredo y otros [139], describen el pensamiento computacional como aquellas fases del pensamiento involucradas en la formación de problemas y búsqueda de soluciones, y que estas soluciones sea posible de implementar por medio de agentes de información para el respectivo procesamiento (ya sea un ser humano, una computadora o una combinación de ambos). Recientemente, García-Valcárcel Muñoz-Repiso y Caballero-González [63], exponen que el pensamiento computacional es una capacidad para resolver problemas con la base de la programación y los fundamentos de las ciencias computacionales. Esta definición, aunque limitada a la informática, resume la gran variedad de conceptos dados al pensamiento computacional.

Actualmente, los estudiantes inmersos en las carreras de informática deben alcanzar un alto nivel de lógica de programación, al igual que el pensamiento computacional, que les permita realizar de manera adecuada la descomposición y resolución de problemas, poniendo en práctica conceptos de algoritmos. En las asignatura contempladas en los primeros semestres de las carreras de

informática, los estudiantes cursan asignaturas de programación en las cuales es necesario que deben desarrollar habilidades de resolución de problemas.

Para Voon y otros [164], el desarrollo de habilidades del pensamiento computacional es una práctica de alfabetización científica asociada a la resolución de problemas, además presenta cinco dimensiones del pensamiento computacional, como son, diseño algorítmico, descomposición, abstracción, evaluación y generalización, estos principios van a permitir a los estudiantes desarrollar las competencias del pensamiento computacional, a saber, el pensamiento crítico, la resolución de problemas, la cooperatividad, el pensamiento algorítmico y la creatividad. Por lo tanto, los individuos deben explorar y adquirir competencias de pensamiento computacional para funcionar eficazmente en el mundo digitalizado.

En la Tabla §2.1, se expone de manera resumida las definiciones de pensamiento computacional, a manera de organizar las ideas para su contextualización en la presente investigación.

Para resumir, se considera que el pensamiento computacional se puede definir como un proceso que consiste en identificar y analizar un problema, establecer posibles soluciones por medio del uso de herramientas tecnológicas, en la cual los participantes desarrollan la lógica, diseñan sistemas y elaboran soluciones mediante una de las alternativas mencionadas para desarrollar el pensamiento computacional, como es la programación. Además, integrando las habilidades de pensamiento computacional y las técnicas de programación, los estudiantes que aprender a programar pueden ofrecer soluciones óptimas y de calidad. Se coincide con [173] con respecto a la definición de pensamiento computacional, además de deducir que desarrollar el pensamiento computacional permite a los estudiantes ofrecer soluciones a las situaciones del mundo real en un mundo virtual con el uso de la tecnología.

2.3. Competencias del pensamiento computacional

A finales de la década de 1970 surgió la idea de que los algoritmos son la base para la informática y la programación, extendiéndose el interés hasta los sistemas y la experimentación. Sin embargo, algunas iniciativas de pensamiento computacional moderno enfatizan el lado de la programación y el algoritmo casi dejando a un lado la experimentación y los sistemas informáticos [152].

El pensamiento computacional es un dominio de competencia de amplia aplicación, que es importante para que los individuos tengan éxito en la sociedad tecnológica actual, para aumentar el interés en las TIC y para apoyar la investigación en otras disciplinas [182].

Autor	Año	Definición
Aho [3]	2012	proceso del pensamiento implicado en la planteamiento de problemas, pudiendo representar las soluciones en algoritmos computacionales.
Bocconi *et al.* [24]	2016	ideas y conceptos clave del campo disciplinario de informática y ciencias de la computación.
Buitrago *et al.* [28]	2017	forma de abordar situaciones cotidianas y resolver problemas utilizando conceptos fundamentales para la informática.
García-Valcárcel *et al.* [63]	2019	capacidad para resolver problemas utilizando la programación y los fundamentos de las ciencias computacionales.
Lee *et al.* [93]	2011	conceptualización, conocimiento y resolución de problemas, razonamiento en diferentes niveles de abstracción, aplicación de la automatización, y análisis de lo idóneo de las abstracciones efectuadas.
Rojas *et al.* [130]	2016	proceso de pensamiento que ofrece soluciones a los problemas por medio del desarrollo de destrezas como la descomposición, la abstracción, la evaluación, la generalización y el diseño de algoritmos.
Selby *et al.* [143]	2014	actividad basada en el cerebro que permite resolver problemas, comprender mejor las situaciones y expresar los valores mediante la aplicación sistemática de la abstracción, descomposición, diseño algorítmico, generalización y evaluación en la producción de una automatización implementable por un dispositivo informático digital o humano.
The British Royal Society [154]	2012	capacidad de identificar características del área de la computación, y determinar que técnicas y herramientas informáticas se deben aplicar para asimilar información concerniente a sistemas naturales y artificiales.
Voon *et al.* [164]	2022	es una práctica de alfabetización científica asociada a la resolución de problemas, con cinco dimensiones: diseño algorítmico, descomposición, abstracción, evaluación y generalización, para desarrollar las competencias del pensamiento computacional: el pensamiento crítico, la resolución de problemas, la cooperatividad, el pensamiento algorítmico y la creatividad.
Voogt *et al.* [163]	2015	contempla procesos del pensamiento que permite la resolución de problemas complejos y la descomposición de este proceso en una diversidad de problemas.
Wing [174]	2006	identificación y resolución de problemas, el diseño de sistemas y la comprensión de la conducta de los individuos por medio conceptos básicos de informática, con la ayuda de herramientas tecnológicas.

Tabla 2.1: Definiciones de pensamiento computacional

Barr y Stephenson [17] ampliaron las ideas iniciales y propusieron nueve conceptos y capacidades centrales de pensamiento computacional, que incluyen actividades de recopilación, análisis y representación de datos, descomposición de problemas, abstracción, algoritmos y procedimientos, automatización, paralelización y simulación.

Para Bocconi y otros [24], aunque en la literatura de pensamiento computacional se cuenta con una variedad de definiciones y propuestas, un subconjunto de conceptos y habilidades centrales emerge recursivamente. Este compendio de competencias, que se presentan en la Tabla §2.2, se pueden resumir en capacidad de abstracción, automatización, descomposición, evaluación y generalización. En cuanto a la abstracción, Wing [173] señala que, es una de las habilidades centrales que los estudiantes deben adquirir en el pensamiento computacional; mientras que Cetin y Dubinsky [32], muestran que ésta puede usarse en el contexto del pensamiento computacional como concepto central.

En la Tabla §2.2, los autores [10, 17, 67, 93, 142] presentan un resumen de los conceptos y capacidades básicas del pensamiento computacional. Los autores interpretan que el proceso de pensamiento computacional inicia con el análisis y formulación de un problema en búsqueda de una solución considerando actividades como la abstracción, descomposición, pensamiento algorítmico, depuración, simulación y generalización.

Por otro lado, de manera general, las diferentes tecnologías que aparecieron en los últimos años usadas en el desarrollo de habilidades para enseñar y aprender pensamiento computacional se pueden organizar en cinco categorías amplias, según [108]:

1. **Actividades desconectadas.** Incluyen juegos de lógica, cartas, rompecabezas, cuerdas o movimientos físicos, para introducir conceptos informáticos (algoritmos, transmisión de datos); evitando el uso de dispositivos digitales, ya que se centran en ideas y conceptos, y no en su implementación.
2. **Entornos visuales basados en flechas.** Los participantes crean programas visualmente usando una sucesión de comandos con flechas o íconos, intuitivos de usar.
3. **Entornos visuales basados en bloques.** Los alumnos programan visualmente, aplicando una serie de comandos de forma sucesiva utilizando bloques o piezas de código que se acoplan.
4. **Lenguajes de programación textual.** Los participantes deben escribir los comandos y considerar la sintaxis específica del programa.
5. **Conexión con el mundo físico.** Los participantes desarrollan programas que controlan y tienen efectos en objetos del mundo físico.

Barr *et al.* [17]	Lee *et al.* [93]	Grover *et al.* [67]	Selby *et al.* [142]	Angeli *et al.* [10]
Abstracción	Abstracción	Abstracción y generalización de patrones	Abstracción	Abstracción
Algoritmos y procedimientos		Nociones algorítmicas de flujo de control	Pensamiento algorítmico	Algoritmos (incluida el flujo de control)
Automatización	Automatización			
	Análisis			
		Lógica condicional		
Descomposición del problema		Descomposición estructurada de problemas (modularización)	Descomposición	Descomposición
		Depuración y detección sistemática de errores		Depuración
		Restricciones de eficiencia y rendimiento	Evaluación	
			Generalizaciones	Generalización
		Pensamiento iterativo, paralelo recursivo		
Paralelización				
Simulación				
		Procesamiento sistemático de la información		

Tabla 2.2: Conceptos y habilidades del pensamiento computacional en la literatura [24], p.17

Generalmente, en las carreras de informática en un nivel universitario se utilizan los lenguajes de programación textual (Java, C, Python) como parte del programa de estudios de las asignaturas de programación. Es importante considerar el nivel de lógica de programación desarrollada por los estudiantes para poder dar solución a los problemas presentados. Los profesores deberían considerar que al incrementar habilidades como la lógica y pensamiento computacional, los estudiantes tendrían mejores oportunidades para aprender conceptos de programación. Es necesario mirar en la literatura la pertinencia de las categorías mencionadas por [108] en cuanto al proceso de aprendizaje de la programación.

En resumen, del pensamiento computacional se derivan competencias y habilidades como la abstracción, algoritmos, generalización, entre otros, las cuales están estrechamente enlazadas con la programación y el desarrollo de habilidades cognitivas, que los estudiantes deben desarrollar.

2.4. Implicaciones en el contexto educativo

Como lo concluyen Voogt y otros [163], la investigación sobre la incorporación del pensamiento computacional en el aula de clases es aún escasa; en particular, existe la necesidad de estudiar cómo se puede desarrollar en estudiantes en disciplinas distintas a la informática. Además, la afirmación de que su desarrollo también aumenta la capacidad de los estudiantes para poder lidiar con la complejidad y los problemas abiertos debe ser estudiada en profundidad. Y, agrega que, se necesita un conocimiento más profundo para estudiar el papel de la programación como herramienta en el desarrollo del pensamiento computacional.

Hoy en día, el pensamiento computacional influye en las áreas vinculadas con las ciencias naturales y sociales, facilita la obtención de habilidades para resolución de problemas, diseñar sistemas y comprender los límites y el poder de la inteligencia humana y de máquinas [28]. El pensamiento computacional influye en las áreas vinculadas con las ciencias naturales y sociales y facilita la adquisición de habilidades para resolver problemas, diseñar sistemas y comprender el poder y los límites de la inteligencia humana y de las máquinas.

Es importante mencionar que existe una brecha crítica en términos de investigación y educación enfocada en el proceso de enseñanza-aprendizaje del pensamiento computacional y la programación; a pesar de la gran cantidad de artículos escritos por científicos informáticos que investigan temas relacionados con la enseñanza y el aprendizaje del pensamiento computacional y la programación, la investigación realizada por especialistas en el área de la educación es casi inexistente en la mayoría de los países [28]; lo que representa una oportunidad para desarrollar iniciativas en el contexto educativo ecuatoriano que apunten en este estudio.

Adell y otros [2], señalan que el pensamiento computacional ha sido considerado en el currículo de muchas instituciones educativas en los últimos años de manera obligatoria en varios países, a pesar de ello, según los expertos no hay suficiente evidencia sobre el marco conceptual con una clara definición con respecto a sus principales elementos. Por su parte Ilic y otros [75], presentan sus principales hallazgos sobre la posición del pensamiento computacional en los currículos y la educación, entre los cuales se encuentran, que este debe integrarse a la educación, que los planes de estudio deben reorganizarse en

función de sus competencias, que es útil en cursos, y que debe incluirse en la formación del profesorado.

A medida que aumenta la demanda de estudiantes en las carreras de informática, es necesario aplicar lineamientos que ayuden al estudiante a desarrollar la lógica y el pensamiento computacional, que permitan a los estudiantes ofrecer soluciones informáticas mediante la resolución de problemas a través de la programación. Esto podría lograrse con el uso de herramientas de programación visual, que minimicen la codificación y sintaxis de los lenguajes de programación tradicionales.

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.

Yadav y otros [182], sugieren examinar cuidadosamente el papel del pensamiento computacional en varias áreas que van mas allá de la práctica de la programación; señalan que, el primer paso en este camino podría ser acordar una definición o, al menos, elaborar una base común de las conceptos principales que sean básicos en todas las disciplinas; y cuyo esfuerzo permitiría a los educadores e investigadores enfocarse en cómo las diferentes construcciones del pensamiento computacional se relacionan con otras disciplinas e integrarlas en su campo.

Finalmente, como lo plantean Aranda y Ferguson [12], se necesita conceptualizar y aplicar el pensamiento computacional como una forma de razonamiento que los estudiantes puedan usar para resolver los problemas que enfrentan como ciudadanos de esta era digital en la que viven actualmente.

2.5. Resumen

El pensamiento computacional ha sido objeto de discusión y divulgación, tal como se evidencia en la extensa literatura del tema. La variedad de conceptos presentados, lejos de presentar un problema, la cuenta de lo importante de este concepto en el quehacer científico e investigador. El pensamiento computacional se puede definir como un proceso de resolución de problemas para la toma de decisiones considerando los recursos de ciencias de la computación; sin embargo, esta definición deja abierta su adaptación a diferentes contextos.

Los cambios recientes que se están generando en la era digital actual, y los que están por emerger, dejan planteadas muchas implicaciones para el ecosistema educativo, que incluyen el establecimiento de una definición formal, el

desarrollo de competencias y habilidades, así como el diseño de un plan de formación para los profesores.

Capítulo 3

Herramientas para el aprendizaje de la programación

Hay solo dos clases de lenguajes de programación: aquellos de los que la gente está siempre quejándose y aquellos que nadie usa.

Bjarne Stroustru

El progreso de la ciencia y la tecnología está modificando el ecosistema actual. En este escenario, se hace imprescindible que los profesionales de las diferentes actividades económicas y sociales adquieran habilidades para enfrentar este nuevo contexto con un enfoque de pensamiento computacional; así, el aprendizaje de la programación se ha evidenciado que es una habilidad esencial que deben desarrollar de manera imperativa los profesionales de ciencias de la computación. El propósito de éste capítulo es identificar las herramientas mas utilizadas para el aprendizaje de la programación.

En la Sección §3.1 se especifica una introducción de las herramientas para el aprendizaje de la programación; más adelante en la Sección §3.2 se describe el uso de las herramientas en un contexto educativo. Luego, en la Sección §3.3 se menciona varias experiencias de los lenguajes de programación educativos en un contexto académico. En la Sección §3.4 se muestran algunos lenguajes de programación educativos. Finalmente, resumimos este capítulo en la Sección §3.5.

3.1. Introducción

En una sociedad donde las TIC afectan a todos los segmentos sociales y donde el aprendizaje, el trabajo y el entrenamiento tienen lugar a un nuevo entorno virtual, se visualiza un potencial crecimiento de un escenario como es la sociedad informática como parte de una sociedad del conocimiento [149]; estas TIC han venido desempeñando un papel importante en el área educativa y en su estructura al plantear nuevas discusiones y desafíos [135]. El desarrollo de la ciencia, la tecnología y la innovación han cambiado todas las condiciones de la sociedad, ya que en el sistema educativo también se necesitaba una forma más moderna y activa de organizar el proceso de enseñanza [43].

La alfabetización mediática se ha vuelto cada vez mas importante, debido a que la sociedad moderna necesita usuarios que desarrollen aptitudes con relación al uso de la tecnología [25]. Así, los esfuerzos de educación en informática se están expandiendo en todo el mundo para introducir habilidades informáticas a los estudiantes para su desarrollo en el mundo digital de hoy [181]. Algunos de los países en vías de desarrollo intentan mejorar la calidad de la educación en nivel superior, integrando el uso de las TIC en la enseñanza, en cuanto a preparación de materias, evaluación y recursos; sin embargo, los resultados no evidencian un uso transformador de las TIC de tal manera que se haga un cambio en los enfoques existentes del proceso de enseñanza dentro del aula [59].

Definitivamente, las TIC son un referente para promover las actividades de aprendizaje en los centros educativos; su adecuada implementación puede incentivar el aprendizaje de competencias esenciales para un desempeño apropiado de un individuo en el campo laboral, personal y social [118]. Las nuevas TIC afectan directamente los métodos, prácticas y recursos de enseñanza, lo cual se traduce en enormes desafíos para la enseñanza tradicional [38].

Actualmente, la escasez de personal en Ciencia, Tecnología, Ingeniería y Matemáticas *(Science, Technology, Engineering and Mathematics - STEM)* está muy extendida [140]. En general se requiere mas programadores de sistemas debido al surgimiento de nuevas tecnologías y a su creciente desarrollo en una era digital, con nuevos retos como el internet de las cosas, hace que surjan nuevas necesidades de aplicaciones [62]. El área de desarrollo de software ha superado una rápida y creciente expansión; por lo que requiere que cada programador deba aprender continuamente y dominar las nuevas tecnologías [105].

Esta situación representa un desafío para la sociedad moderna al preparar a las nuevas generaciones de desarrolladores de software, ya que requiere personas con conocimientos en algoritmos y programación de computadoras [26]. Afortunadamente, una parte esencial de los estudios en las facultades que prepa-

ran a los estudiantes en el área de ciencias de la computación, es el desarrollo de la capacidad de los estudiantes para pensar algorítmicamente [105].

Por otro lado, autores como Davis [46], sostienen que, las herramientas digitales han aumentado el costo de la educación necesaria para el mundo digital y aumentado la complejidad de la enseñanza. Otros, como Parra Sarmiento y otros [118], afirman que, a pesar de las iniciativas a favor del desarrollo de las TIC en el ámbito educativo, algunos factores inciden en su implementación como son formación de docentes, gestión de TIC, recursos económicos y la poca motivación del uso de las TIC.

Actualmente, una de las principales tendencias en el panorama educativo a nivel mundial, es la inclusión de habilidades de programación informática y pensamiento computacional en el currículo escolar [108]. Para Li [97], el aprendizaje de programación es una forma efectiva de cultivas la capacidad de pensamiento computacional; así, existe un creciente interés por la forma en que se debe implementar el pensamiento computacional en el enfoque de enseñanza, especialmente para estudiantes sin uso previo de computadoras [121]. Se hace necesario entonces, identificar las mejores prácticas en desarrollo de herramientas para el aprendizaje de programación, con el propósito de sustentar cualquier iniciativa en beneficio de la generación de nuevos recursos que ayuden al desempeño, tanto a los docentes en su rol de enseñanza, como a los estudiantes en sus actividades de aprendizaje.

3.2. El uso de herramientas informáticas en el contexto educativo

En esta etapa que finaliza la segunda década del siglo XXI, se está presentando un repentino crecimiento en el área de ciencia y tecnología, por consiguiente en el campo de la educación como resultado visible de la revolución científico-tecnológica que brinda nuevas oportunidades [149]. Entre las oportunidades del uso de computadoras en el campo de la educación, los programas educativos han aparecido masivamente en el mercado [38].

Como ya lo anunciaban como tendencia tecnológica Armstrong y Loane [13], el software educativo es un segmento que ha estado en evidente crecimiento; ha beneficiado a los estudiantes en el proceso de aprendizaje de manera indiscutible desde que emergieron a mediados de los noventa. Roschelle y otros [131], también lo proyectaba cuando señalaba que, la demanda de software educativo estaba creciendo exponencialmente con el aumento del interés en la reforma educativa, Internet y el aprendizaje a distancia; y que, las aplicaciones educativas debían ser flexibles porque los currículos y los estilos de

enseñanza variaban entre las instituciones, las ubicaciones e incluso entre los instructores de la misma institución. Por su parte, Pellone [119] a mediados de los años 90's, ya reportaba una revisión de literatura donde delineaba el ámbito de la teoría en el diseño de software educativo para la educación y capacidad vocacional, y daban cuenta de la importancia de los "ingredientes clave"que constituyen las mejores prácticas.

Por su parte, Cohen y otros [37], presentaban un método para evaluar la efectividad del software educativo, y discuten varios temas, incluido el uso de un entorno experimental frente a un estudio de campo y el diseño de instrumentos de evaluación adecuados para la evaluación de software educativo. Así mismo, Huber y Giuse [73], documentan un proceso de evaluación de software educativos, usando un enfoque de equipo, empleando un instrumento de evaluación de dos niveles basado en formularios de evaluación de software existentes e informes de errores del sistema.

Los juegos educativos son una estrategia de instrucción efectiva y eficiente para la educación en computación, por lo que, es esencial evaluar sistemáticamente tales juegos para obtener evidencia sólida de su impacto [120]. Dado el interés que los investigadores han mostrado en la aplicación de juegos en ingeniería de software, Kosa y otros [91], realizan una RSL de la educación en ingeniería de software y juegos; encontrando como principales hallazgos que los estudios se acumulan en 5 categorías: juegos que los estudiantes usan, juegos que los estudiantes desarrollan como proyectos, propuestas curriculares, desarrollo/creación de nuevos enfoques, herramientas, marcos o sugerencias y otros. Así, Petri y Gresse von Wangenheim [120], presentan el estado del arte sobre cómo se evalúan los juegos para la educación informática, encontrando que la mayoría de las evaluaciones utilizan un diseño de investigación simple, tamaños de muestra pequeño, enfoque cualitativo para el análisis de datos y no utilizan un modelo o método de evaluación definido; lo que demuestra que se necesitan evaluaciones más rigurosas y apoyo metodológico para ayudar a los creadores de juegos e instructores a mejorar dichos juegos, así como a apoyar sistemáticamente las decisiones sobre cuándo o cómo incluirlos en las unidades de instrucción.

So y Seo [148], por su parte, documentan una RSL sobre el aprendizaje basado en juegos y la investigación de la gamificación en Asia; aunque, los hallazgos defendieron los efectos positivos de los juegos en los resultados de aprendizaje, se identifican algunas brechas de investigación, incluida la falta de diversidad en las disciplinas temáticas y géneros de juegos, el predominio de los experimentos de comparación de medios y los problemas de sostenibilidad y escalabilidad. Más recientemente, Lino Ferreira da Silva Barros y otros [98], presentan una RSL, para proporcionar una bibliografía de referencia para la construcción de objetos de aprendizaje multimedia aplicados a las platafor-

mas de Stewart utilizando Métodos de ingeniería de software, aportando resultados exitosos y publicaciones relevantes con factores de alto impacto, lo que constituye una base sólida para varios trabajos en los campos de investigación.

3.3. Experiencias sobre la enseñanza de programación

En educación informática, se reconoce ampliamente que las actividades de programación son cruciales para el buen desempeño de los estudiantes; de hecho, para Moreno-León [108], la manera mas efectiva de fomentar el pensamiento computacional desde edades tempranas es incluir actividades de programación en el currículo escolar. Para Janpla y Piriyasurawong [78], desarrollar programas de computadoras es un talento necesario en ciencias de la computación. Sin embargo, cualquier enfoque que intente predecir este éxito basándose únicamente en las actividades de programación, puede pasar por alto otros factores críticos para su éxito o fracaso dentro de la disciplina [31].

El software educativo tiene la oportunidad de integrar multimedia e interacción tanto para estudiantes como para profesores. Las prácticas educativas también deben enfatizar la creación de entornos de aprendizaje, en el que los alumnos desarrollan sus conocimientos y el maestro guía y alienta el proceso. El análisis de los recursos que se traen a través de estas nuevas tecnologías es de suma importancia para capturar, tratar, organizar, sistematizar, conservar y transmitir la información de acuerdo con los objetivos pedagógicos intrínsecos [38].

En cuanto al aprendizaje de programación, este proceso debe considerar como experiencia positiva; sin embargo, son muchos los problemas con los que se encuentran los estudiantes cuando se inician con esta actividad [77]. En general, las herramientas educativas para programación en informática (incluidos los Educational Programming Languages (EPLs) y otros entornos educativos) se pueden utilizar para enseñar y desarrollar habilidades de programación en niños, jóvenes y adultos. En su mayoría, las herramientas se basan en lenguajes de programación diseñados para ser utilizados como recursos de aprendizaje, que generalmente se basan en bloques y contienen elementos básicos requeridos en la programación tradicional.

3.4. Los entornos de programación visual basados en bloques

Los entornos de programación visual basados en bloques, usados hoy en día por millones de estudiantes [144], son una buena estrategia para la introducción, inmersión y participación a los nuevos programadores en el mundo de la programación informática [180]; [27]. Aunque este enfoque no es una innovación reciente, se ha generalizado en los últimos años con el surgimiento de una nueva generación de herramientas diseñadas bajo un entorno basado en bloques [171].

Aunque los bloques aparecen en editores estructurados desde inicios de los noventa [106], el concepto de entorno de programación visual por bloques, fue introducido por primera vez en 1996 por el MIT Media Lab, con *Logo Blocks*, versión gráfica de los ladrillos programables controlados por el logotipo de Cricket - pequeños microcontroladores que se usaron en proyectos de robótica [153].

Las herramientas basadas en bloques combinan una gran variedad de funciones para producir entornos de programación accesibles y atractivos [27]. Los bloques son representaciones visuales de código de programa que agregan forma, color y edición de arrastrar y soltar a la sintaxis concreta; ofrecen una serie de posibilidades como prevención de errores de sintaxis, sugerencias de tipo y descubrimiento de APIs. [144]. A continuación, se mencionan (orden alfabético) algunos entornos de programación visual basados en bloques y utilizados por los investigadores considerados en este estudio.

1. ***AgentCubes***: creado por *AgentSheets*, es un entorno de programación con herramientas intuitivas, visuales e integradas basadas en agentes, simulaciones y juegos en 3D y que permite compartir las experiencias en línea. Este recurso facilita formación de aptitudes creativas para la resolución de problemas y la lógica de programación. *AgentSheets* combinó cuatro opciones clave para crear una forma inicial de programación de bloques [125].

    - Los bloques son compensables por el usuario final.
    - Los bloques son editables por el usuario final.
    - Los bloques se pueden anidar para representar estructuras de árboles.
    - Los bloques están dispuestos geométricamente para definir la sintaxis.

    Entre alguna experiencias con *AgentCubes*, se destacan Leonard y otros

[95], quienes reportan el aprendizaje por parte de los profesores para facilitar el pensamiento computacional entre sus estudiantes; Ioannidou y otros [76], quienes describen y evalúan la idea de incrementar 3D como un enfoque para que los usuarios finales creen juegos 3D y adquieran fluidez; entre otros. Así desde su inicio, *AgentCubes* se ha convertido en una herramienta de pensamiento computacional altamente accesible [125].

AgentSheets, de los mismos creadores de *AgentCubes*, es una herramienta para el desarrollo del pensamiento computacional basada en agentes que facilitan a los muchos usuarios finales (desde niños hasta profesionales) crear sus propios juegos y ampliaciones de ciencia computacional (Manual References). *AgentSheets* es una herramienta para construir aplicaciones gráficas interactivas; combina la facilidad de uso de un kit de construcción con la flexibilidad de un entorno de programación visual [126].

2. ***ALICE***: desarrollado en la Universidad Carnegie Mellon, considerado un lenguaje de programación innovador basado en bloques, permite la creación de animaciones o narraciones interactivas y desarrollo de juegos 3D simples. El software ALICE está diseñado para preparar a los participantes en desarrollo de habilidades de pensamiento lógico y computacional, principios básicos de programación y utilizado como fase de introducción a la programación orientada a objetos. Además, tiende a orientarse hacia los estudiantes más jóvenes.

   Algunas experiencias con ALICE, incluyen a Chang [33], quien evalúa los efectos de su uso con la introducción de la programación; Dwarika y De Villiers [51], quienes la usan como estrategia para desarrollar la comprensión de los participantes en la programación; Costa [41], quien presenta un diseño de entorno de aprendizaje que compara la efectividad del uso de este software con los lenguajes de programación convencionales en contexto del aprendizaje de programación inicial; entre otros.

3. ***AppInventor***: creado por Google Labs, es un lenguaje de programación intuitivo y visual, es útil para crear aplicaciones totalmente funcionales para teléfonos inteligentes y tabletas. Es un entorno de programación basado en bloques de fácil uso para la creación de aplicaciones complejas y funcionales, sin la necesidad lidiar con una sintaxis complicada y en mucho menos tiempo que los entornos de programación tradicionales.

   AppInventor, es un entorno de aprendizaje en línea informal con aproximadamente 6.8 millones de usuarios y 22 millones de proyectos / aplicaciones creadas[†1].

[†1] https://appinventor.mit.edu/about-us

Entre las experiencias con *AppInventor*, se destacan Kang y otros [84], quienes estudian una aplicación con *Android Application Prototyping Method* usando *AppInventor*; A-Ghamdi y otros [1], quienes usan la aplicación Android para móviles en un taller de Programación; Rizzo y otros [127], quienes con estudios de casos describen las oportunidades que brinda *AppInventor* en cuanto a las mejoras en la experiencia del proceso de aprendizaje de programación; Xie y Abelson [179], explora cómo las personas desarrollan habilidades computacionales mientras crean con *AppInventor*; entre otros.

4. ***PiktoMir***: sistema de programas de distribución gratuita, es útil para aprender los principios básicos de la programación en niños de preescolar y primaria. Se enfoca principalmente en preescolares que aún no pueden escribir, o para estudiantes de secundaria que no dominan la escritura.

   Entre las experiencias con la plataforma *PiktoMir*, se destacan Besshaposhnikov y otros [21], quienes desarrollan un sistema educativo y de juegos para niños preescolares; Rogozhkina y Kushnirenko [129], quienes investigaron la viabilidad de usarla en la enseñanza de elementos de programación para niños de edad preescolar; entre otros.

5. ***Scratch***: es un entorno de programación donde se pueden crear historias interactivas, animaciones y juegos. Es útil para desarrollar el pensamiento de manera creativa, aprender a razonar sistemáticamente e incluso trabajar en colaboración. *Scrtach* es un proyecto apoyado por Scratch Foundation desde el 2013[†2].

   Entre las experiencias con *Scrtach* se destacan, Chang y otros [34], quienes diseñan e implementan una nueva herramienta de análisis de este programa para evaluar las habilidades de pensamiento computacional; Weintrop y otros [170], quienes reportan las experiencias en la introducción de la programación; y, Erol y Kurt [54], quienes examinan le efecto de la enseñanza de programación con Scratch en la motivación y desempeño de los estudiantes; entre otros.

6. ***Snap***: es un entorno de programación visual. Es una versión extendida de Scratch que permite diseñar y construir tus propios bloques. Presenta nuevas características añadidas como listas y procedimientos, lo que hace que Snap sea apropiado para una introducción avanzada a la informática para estudiantes de secundario o universidad. Es un proyecto del Grupo Lifelong Kindergarten del MIT Media Lab[†3].

[†2] https://www.scratchfoundation.org/our-story
[†3] https://snap.berkeley.edu/

Entre los autores que reportan sus experiencias con Snap, se encuentran Leskovec y Sosič [96], quienes describen el potencial de la herramienta; entre otros.

También se consideraron algunos lenguajes de programación textual utilizados en trabajos analizados en el presente estudio, y que su enfoque está dado en el método de enseñanza a través de dichos lenguajes de programación textual. A continuación se presentan brevemente algunos de ellos:

1. ***Lenguaje C***: es un lenguaje de programación estructurado y utilizado para desarrollar editores, compiladores, sistemas operativos y aplicaciones potentes; sin embargo, no es sencillo aprender a programar en C por su compleja sintaxis.

   Entre las experiencias reportadas está Kordaki [90], donde se presentan resultados positivos de un estudio piloto para el aprendizaje de la programación en C por parte de principiantes; por otra parte, Tuparov y otros [156] reportan el incremento del interés de los participantes en la programación desarrollando algoritmos usando C++, que es una extensión del lenguaje C que permite la utilización de objetos.

2. ***Java***: es un lenguaje orientado a objetos, con el que se pueden desarrollar aplicaciones de muchas áreas, como seguridad, bases de datos, aplicaciones cliente-servidor, páginas web interactivas, aplicaciones móviles, entre otras.

   Entre los autores que reportan su experiencia están Reardon y Tangney [124], Wang y otros [167], que reportan la eficacia a superar las barreras a las que hacen frente los programadores novatos, observando que los estudiantes desarrollaron motivación y compromiso en el aprendizaje de la programación.

3. ***Python***: lenguaje de programación orientado a objetos, utilizado a nivel mundial, principalmente para construir aplicaciones web, automatizar procesos y crear aplicaciones empresariales robustas. Como ejemplos de programas creados con Python están Dropbox, Netflix, Instragram, entre otros.

   Entre los autores que reportan su experiencia están Aycock y otros [15], Lee y Ko [94], en cuyos estudios demostraron que utilizar Python en curso introductorio de programación ayuda a la comprensión de conceptos.

A continuación, se mencionan otras herramientas educativas, que no son lenguajes de programación pero que se han utilizado para la enseñanza de programación, como son los juegos multiplataforma.

1. ***Gidget***: es un juego de computadora, diseñado para enseñar conceptos de programación a través de la depuración de rompecabezas. El trabajo de los jugadores es corregir el código de Gidget para completar todas las misiones del robot.
2. ***Jeliot***: Jeliot I se desarrolló en la Universidad de Helsinki-Finlandia, mientras que Jeliot 2000 se desarrolló en el Instituto Weizman-Israel. Son aplicaciones de visualización de programas, que permiten ver cómo se interpreta paso a paso un programa Java, así como visualizar las llamadas a los métodos, variables y operaciones en una pantalla animada. Se han desarrollado varias versiones de este entorno.

3.5. Resumen

Dentro de las estrategias para desarrollar competencias en el pensamiento computacional se destaca el aprendizaje de la programación, donde, dada las diversas dificultades encontradas en los nuevos profesionales de ciencias de la computación, se han presentado una variedad de herramientas para su aprendizaje efectivo. Los entornos de programación visual basada en bloques son una de las opciones mas usadas y de mayor efectividad para aprender programación y desarrollar el pensamiento computacional; dentro de este grupo que incluye Scratch, AppInventor, entro otras, se destaca la plataforma Alice, cuyas evidencias reportadas la posicionan como alternativa amigable para los nuevos programadores.

Capítulo 4

El entorno de programación ALICE

Las paredes de ladrillo están ahí por una razón. Las paredes de ladrillo no están ahí para impedirnos el paso. Las paredes de ladrillo están ahí para darnos la oportunidad de demostrar lo mucho que queremos algo. Porque las paredes de ladrillo están ahí para detener a las personas que no lo desean lo suficiente. Están ahí para detener a las otras personas.

Randy Pausch, The Las Lecture

El Aprendizaje de programación, es una competencia esencial para los profesionales de esta era, que aunque de naturaleza abstracta, cuenta con muchos recursos y medios para facilitar esta tarea. ALICE es un lenguaje de programación con orientación a objetos para la iniciación en esta difícil experiencia. Desarrollado por la Carnegie Mellon University y actualmente con el respaldo de importantes corporaciones informática, es hoy en día, una de las plataformas de aprendizaje de programación más usadas, particularmente para los que inician en estas actividades.

En la Sección §4.1 se especifica una introducción del entorno ALICE; más adelante en la Sección §4.2 se describe el entorno de ALICE y su enfoque en la programación orientada a objetos. Luego, en la Sección §4.3 se menciona las fortalezas de ALICE en el proceso de aprendizaje de la programación. En la Sección §4.4 se detallan algunas experiencias con el uso de ALICE . Finalmente, resumimos este capítulo en la Sección §4.5

4.1. Introducción

La importancia de la educación en programación informática ha ido creciendo junto con la difusión de las aplicaciones computacionales en muchos aspectos del trabajo y la vida cotidiana [103]. Aprender a programar en código de computadora ha sido considerado como uno de los pilares de la educación contemporánea con beneficios que van mucho mas allá de las habilidades requeridas por la industria de la computación, hacia la creatividad y la autoexpresión [102].

La programación es una destreza elemental que la mayoría de los estudiantes de informática deben adquirir y aprender [55]. Si bien la programación informática es una actividad altamente abstracta y, de hecho, cognitiva, el aprendizaje de la programación informática puede beneficiarse si se canaliza a través de la incorporación de medios y recursos adecuados [102]. Para muchos estudiantes de ciencias de la computación, aprender programación se considera una tarea difícil [9, 20]; pues requiere de aprender también conceptos y lenguajes de programación; además, el usuario encuentra dificultad al visualizar los cambios que están ocurriendo cuando se ejecuta un programa. ALICE puede hacer mas fácil el aprendizaje de programación ayudando a resolver estos problemas.

La corrección del código de programación es otro aspecto del trabajo de los programadores, lo cual hace que las técnicas actuales sean inadecuadas para analizar y diseñar los sistemas que los programadores utilizan, ya que las mismas se centran más en problemas con la capacidad de aprendizaje y la eficiencia de uso y menos en la tendencia a errores [89]. La sintaxis para la programación de computadoras es considerada rigurosa porque sigue reglas rígidas que no permiten la maniobra ni la desviación; en el entorno ALICE, se centra en la sintaxis del lenguaje, facilitando a los participantes la creación de animaciones y/o juegos [8].

La plataforma de programación ALICE, fue desarrollada en la Carnegie Mellon University por equipo dirigido por Randy Pausch [39]; y fue creado para abordar muchas de las cuestiones planteadas con respecto a la dificultad de los lenguajes de programación, en particular para los nuevos usuarios [8]. La estructura de ALICE proporciona un sistema de programación moderno y potente que admite métodos, funciones, variables, parámetros, recursos, matrices y eventos.

ALICE ha aumentado en popularidad para su uso en los cursos de programación de primer año tanto en universidades como en escuelas secundarias; esta creciente popularidad como primer lenguaje de programación, se debe a las muchas ventajas que brinda sobre las alternativas tradicionales [8]. En el entorno universitario, la implementación de ALICE ha tomado muchas for-

mas, desde módulos para comenzar las clases de programación hasta cursos semestrales completos que enseñan conceptos fundamentales de programación usando esta plataforma [138]. Así, ALICE es a la vez una herramienta de software innovadora y con un enfoque pedagógico diseñado para permitir que los conceptos de programación tradicionales sean mas fáciles de enseñar y mas fáciles de entender por los estudiantes de hoy [64].

ALICE proporciona un entorno de programación animada en 3D que admite una perspectiva para enseñar la resolución de problemas de la manera particular utilizada en la programación [39]; fomenta en los estudiantes las habilidades de resolución de problemas y pensamiento lógico necesario en muchos aspectos, por lo que, su uso puede mejorar la capacidad de comprender el código de los programas, lo que forma parte de la base necesaria para facilitar la escritura del código de los programas [141]. ALICE permite a los participantes ver de inmediato la forma cómo se ejecutan sus programas de animación, lo que permite un entendimiento fácil de la relación entre los estados y comportamiento de los objetos [65].

El campo de los juegos y el aprendizaje ha sido explorado en la última década, pero la mayor parte de este trabajo se centra en el juego, en lugar de su diseño; un considerable número de herramientas de creación de juegos gratuitas y fáciles de usar para principiantes ha llevado a un mayor interés en los beneficios educativos de la programación de juegos de computadora [47]; aquí es donde ALICE, se ha presentado como herramienta de soporte.

4.2. El entorno de programación ALICE

ALICE es un lenguaje de programación visual que permite crear programas con el uso de animaciones, narraciones, entornos 3D y gráficos 3D interactivos. Con ALICE se pueden crear historias o juegos interactivos sin la necesidad de entender la sintaxis de un lenguaje de programación concreto, ya que tiene una interfaz que permite arrastrar y soltar objetos, indicando qué partes forman el código, qué se ejecuta y en qué orden, componiendo así el código completo a ejecutar, y facilitando la programación. En la Figura §4.1 se presenta la interfaz de ALICE.

El entorno de programación ALICE puede utilizarse para crear 1) animaciones: usando objetos tridimensionales para crear programas animados, 2) proyectos: creando objetos que se pueden compartir en otros proyectos; y, 3) juegos: deben seguir un enfoque diferente al tradicional para obtener los efectos esperados. Sin embargo, considerando el entorno 3D de ALICE, otros críticos no están de acuerdo en que sea una herramienta que se pueda utilizar para enseñanza de la programación empresarial [138].

Recuperado de http://www.schoolfreeware.com/Freeware_Alice_Download.html

Figura 4.1: Interfaz de ALICE

ALICE es una herramienta de enseñanza, tiene versiones disponibles para Windows y Mac OS X 10.4 o superior.

4.2.1. ALICE como lenguaje orientado a objetos

Durante mucho tiempo, los contenidos de los programas de estudio de ciencias de la computación han hecho una transición a lenguaje C y luego a lenguajes orientados a objetos [55]. El paradigma de la orientación a objetos es la construcción fundamental para los lenguajes de programación modernos, y tanto enseñar como aprender este paradigma son tareas difíciles en el mejor de los casos; sin embargo, las ilustraciones, el uso de gráficos, diagramas, dibujos animados, etc. son mas efectivas que el texto excesivo [64].

Utilizar objetos en un ambiente virtual, hace que los participantes adquieran experiencia en casi todos los aspectos de programación que normalmente se enseñan en un curso de programación introductorio [65]. Los entornos de programación basados en bloques se han utilizado de manera efectiva como poderosas ayudas educativas para introducir a los principiantes en la computación [57]; aquí es donde, ALICE, emerge como herramienta eficaz para adquirir conceptos de programación.

ALICE pretende ser una primera exposición en lenguajes orientados a objetos, pero protege a los estudiantes de consideraciones tales como la sintaxis o el enlace complejo de eventos [168]. ALICE utiliza gráficos y animaciones en 3D y una interfaz de arrastrar y soltar para ofrecer una primera práctica mas significativa en el aprendizaje; en la interfaz interactiva de ALICE, los estudiantes arrastran y sueltan mosaicos gráficos creando programas y generando instrucciones estándares en lenguajes de programación orientados a la producción como Java, C# y C++ [65].

4.3. Acerca de las fortalezas de ALICE

Las ventajas de usar ALICE como plataforma para el proceso de aprendizaje de introducción a la programación, han sido reseñadas por muchos autores. En este sentido Goulet y Slater [64] señala que ALICE pretende proporcionar una forma para ayudar a vencer cuatro obstáculos principales encontrados por los participantes principiantes en los primeros cursos de programación:

1. Los riesgos en la creación de programas, particularmente en la navegación por lo complejo de la sintaxis de los lenguajes de programación.
2. La dificultad de dominar las herramientas complejas (entornos de programación) utilizadas en la creación de programas en lenguajes de programación modernos.
3. La dificultad de ver el proceso en curso y los resultados intermedios de cálculo mientras se ejecuta el programa.
4. La falta de un conjunto de problemas rico, motivador e interesante para estimular la curiosidad del alumno y el deseo de dominar las habilidades necesarias para la creación de programas.

Por su parte Ali y Smith [8], resumen que la ventaja fundamental de enseñar ALICE como primer curso de programación es que aborda los cuatro situaciones que hacen que aprender a programar sea una tarea difícil. En primer lugar, resuelve el problema de la sintaxis; segundo, permite a los programadores ver los resultados en cualquier momento durante el desarrollo y al finalizar sus programas; tercero, aborda la falta de motivación para aprender a programar, considerada como una de las razones para alejar a los estudiantes de los cursos de programación; y, finalmente, reduce el tiempo de desarrollo en comparación con otros lenguajes de programación populares.

Con base a la experiencia de implementación de ALICE, Jones y otros [80] señalan que, el aspecto mas convincente del uso de esa herramienta para la escritura creativa es la inmediatez del texto tridimensional que representa; don-

de los estudiantes experimentan con diferentes personajes y escenarios, pero a diferencia de la historia escrita, en ALICE, pueden visualizar la historia, y cuando incorporan la acción, el programa los obliga a considerar la trama y el impacto emocional y cómo hacer el diseño/codificarlo; por lo tanto, deben pensar en el siguiente paso y simultáneamente editar y volver a crear.

Agregan Goulet y Slater [64] que, además de las fortalezas tangibles de presentar conceptos orientados a objetos, los beneficios intangibles de ALICE incluyen mejores recompensas visuales y refuerzo, motivación intrínseca desde la perspectiva de la narración, confianza en sí mismo y mayor interés en las computadoras. Por otro lado, a pesar de que, se han dedicado mas de 30 años de investigación al desarrollo de sistemas de programación que abren esta actividad a un grupo más amplio de personas, pocos han demostrado formalmente el éxito en niñas de secundaria [86]; algunas experiencias con ALICE, han sido reportadas en este sentido.

Sin embargo, como señala Schultz [138], aunque son muchas las fortalezas de ALICE existen inconvenientes en el uso de la herramienta en el entorno universitario, quien agrega que, debido a que su enfoque es contar historias utilizando objetos visuales en 3D, otros críticos no creen que sea una herramienta adecuada para usar, especialmente al enseñar programación empresarial, porque los ejemplos y los problemas de programación no se prestan a ese entorno. Así mismo, Noshin y Ahmed [115], destaca que, aunque ALICE es un lenguaje animado que no produce ningún error sintáctico, por lo que los estudiantes pueden enfocarse en los conceptos de codificación en lugar de la gramática; la ventaja de usar este tipo de lenguaje es que los estudiantes lo encuentran interesante y fácil, pero la desventaja es que su aprendizaje no es una habilidad empresarial.

4.4. Experiencias con el uso de ALICE

Ali y Smith [8], en el marco de una propuesta de enseñanza de un lenguaje de programación introductorio a nivel universitario, explican las características ALICE, abordando muchas de las cuestiones asociadas con la dificultad de aprender a programar. Como resultado de la experiencia, los autores rediseñaron el curso y el currículum; así mismo, apoyaron la experiencia con una revisión de literatura del tema.

Kelleher y Pausch [86], evalúan ALICE y *Storytelling ALICE*, una versión que introduce la programación en chicas de educación secundaria mediante la creación de historias animadas en 3D; y, demuestran la efectividad de ambas plataformas para aprender construcciones básicas de programación.

Denner y otros [47], usan el entorno de programación de ALICE como herramienta para desarrollar y evaluar competencias de pensamiento computacional en estudiantes de secundaria. El estudio utiliza la mecánica del juego como una forma para pensar. El entorno de programación de los juegos fue codificado para mecanismos que requerían que los programadores pensaran de manera dinámica, dependiente del tiempo o complejas. Se destaca que la mecánica de juego mas común para los estudiantes involucra la programación de eventos, que ALICE requiere la menor cantidad de coordinación entre las piezas del entorno de programación; sin embargo, dentro de las limitaciones del presente trabajo se considera la interfaz de programación de ALICE.

Harrison [70], desarrolla un programa de introducción a la programación utilizando el entorno de programación de ALICE, tomando en cuenta las necesidades de diversidad de los estudiantes, los cambios en la población estudiantil y en el currículo, los antecedentes matemáticos débiles y la necesidad de un curso para cumplir con los requisitos de graduación. Por otro lado, se estaba en la búsqueda de maneras significativas e interesantes de introducir Java a estos estudiantes basándose en los conceptos de informática desarrollados en ALICE. La experiencia se utilizó para generalizar a otras escuelas secundarias en Virginia Beach (USA) que también utilizan ALICE.

Mac Gaul de Jorge y otros [100], estudiaron ALICE, como una herramienta que motiva a los participantes al aprendizaje de la programación en un entorno lúdico. El proyecto estuvo orientado a estudiantes universitarios de Informática, y consistió en desarrollar encuentros bajo modalidad taller. De esta investigación se desprende, que la herramienta ALICE, facilita la articulación vertical entre asignaturas de las carreras; así mismo, y debido a las facilidades que brinda para crear juegos y animaciones con diferentes niveles de complejidad, tanto en lo que se refiere a la programación como el diseño del escenario, es importante definir el alcance que tendrá el juego-animación que se esté desarrollando.

Werner y otros [172], describen un curso de programación de juegos con ALICE en estudiantes de secundaria. Estos autores plantean que, dado que los ambientes de programación que incorporan métodos de arrastrar y soltar y muchos objetos y operaciones predefinidos se están utilizando ampliamente en secundaria, entonces, estos estudiantes podrían aprender conceptos mas complejos de informática utilizando estos entornos de programación cuando la informática no fuera el foco del curso. Así, presentan un análisis de diseños de juegos en los que se miden la frecuencia de ejecución exitosa de construcciones de programación. Los resultados de la investigación evidencian que muchos juegos creados por estudiantes, exhiben usos exitosos de conceptos de informática de alto nivel tales como abstracciones, ejecución recurrente y manejadores de eventos. Los autores discuten las implicaciones de estos resul-

tados para diseñar cursos efectivos de programación de juegos para jóvenes estudiantes.

Schultz [138], documenta la implementación de ALICE para enseñar la lógica de programación de computadoras a estudiantes universitarios. En este estudio, se evaluó la eficacia del entorno de ALICE y se comparó con otra experiencia mas tradicional utilizando diagramas de flujo y pseudocódigo. Aunque, con ambos grupos se reportan altos niveles de satisfacción, el análisis de los datos reveló que no había diferencias estadísticamente significativas. Adicionalmente, el autor discute las maneras de mejorar potencialmente la implementación de ALICE y sus beneficios en el aula.

Jones y otros [80], presentan un enfoque para la enseñanza de un curso interdisciplinario de programación introductoria, usando el sistema de programación de ALICE para enseñar conceptos de programación, diseño e implementación con actividades creativas; demostrando que la creatividad afecta positivamente el diseño y la implementación de un programa. Con respecto a la sensación de los estudiantes que participaron en el experimento, los indicadores reportan resultados positivos. Agregan los autores que, el diseño de los proyectos producidos estaba apropiado y eran mas creativos que el diseño de las ofertas anteriores tradicionales del curso; y que, los participantes eran regularmente mas reflexivos, seguros y articulados con respecto al contenido del curso y los fundamentos que habían aprendido.

Rodger y otros [128], describen la integración de los mundos virtuales de ALICE 3D en muchas disciplinas de la escuela primaria y secundaria; desarrollan una amplia gama de materiales didácticos, que incluye tutoriales para conceptos de informática y conceptos de animación. Para fomentar la construcción de mundos mas complicados, desarrollaron modelos de clases y mundos de ALICE. Con ese material expusieron a profesores y estudiantes a conceptos de computación mientras utilizaban la herramienta ALICE para crear proyectos, historias, juegos y cuestionarios. Los autores destacan la respuesta entusiasta de los profesores de una variedad de disciplinas que asistieron a los talleres de ALICE, los cuales encontraron maneras interesantes de integrar esta herramienta a su tema; crearon tutoriales de ALICE para proporcionar recursos de enseñanza para estos profesores, que van desde principiantes hasta proyectos completos e incluyen conceptos de informática y mejoras de animación.

Ward y otros [168], analizan mediante estudios de casos, la aplicación ALICE y Scratch. Los autores implementaron un algoritmo de clasificación en paralelo y un juego de conversión de números binarios en Scratch y en ALICE. Con la base de sus investigaciones, concluyen que tanto Scratch como ALICE representan una manera innovadora e interesante de enseñar programación, y pudieron usarlos para crear programas aplicando conceptos significativos de ciencias de la computación. Destacan que, el objeto de visualización de matriz

en ALICE es una característica que ayudan a reducir la carga adicional que se produce al agregar aspectos visuales, y que, aunque es menos estable que Scratch, ALICE proporciona mas funciones similares a un lenguaje de uso general, como las verdaderas estructuras de clase y herencia, que tienen estos lenguajes.

Wang y otros [167], realizaron un estudio para establecer si ALICE es una buena opción para enseñar conceptos de programación en estudiantes de nivel secundario. En el experimento se seleccionaron 4 grupos, a 2 se les enseñó programación con ALICE, y a los otros 2, con C++. Los constructos de programación incluyen variables, expresiones aritméticas, estructura de selección, estructura de repetición y funciones incorporadas. Los resultados revelaron que los estudiantes que usaron ALICE tuvieron un mejor desempeño que los estudiantes que usaron C++, lo que indica que ALICE parece ser mas efectivo para facilitar el aprendizaje de los conceptos básicos de programación. Con respecto a la motivación, los estudiantes no se evidenciaron diferencias significativas entre los dos grupos.

Sattar y Lorenzen [137], diseñan e implementan un curso introductorio de programación de computadoras para estudiantes de otras especialidades diferentes a Ingeniería de Software utilizando ALICE, donde dan una visión general de la informática como una disciplina académica y enseñan la programación a esas especialidades de una manera divertida. El curso se organizó en torno a un sitio web y dirigido por un programa de la página web con conferencias interactivas en clase y laboratorios y un proyecto final. Como parte de los hallazgos, los autores destacan que los estudiantes manifestaron disfrutar de trabajar con la herramienta ALICE y particularmente les gustó el proyecto final, ya que ejercitaron su creatividad. Los autores agregan que, cuando midieron pre y post actitudes hacia la informática como una disciplina, encontraron que los estudiantes terminaron pensando que la computación es mucho mas interesante que cuando ingresaron.

Utting y otros [158], discuten sobre los objetivos, mecanismos y efectos de tres entornos: ALICE, Greenfoot y Scratch, los cuales apoyan la inclusión y desarrollo de competencias informáticas (resolución de problemas y programación) en estudiantes preuniversitarios y no técnicos. Entre algunos aspectos resultantes de la discusión, se destaca la agrupación de los usuarios por edad, S1/pre-CS1 para ALICE, 8-16 años para Scratch, y 14+ años para Greenfoot. De la discusión se desprende también que la diferencia 2D/3D es una forma ligeramente diferente de distinguir Scratch y Greenfoot de ALICE, mas como un problema de complejidad.

Graczyńska [65], propone aplicar ALICE como herramienta útil de enseñanza en la secundaria para atraer a los alumnos (especialmente chicas) en actividades como; la introducción a la programación, la realización de videos cortos

con música MP3 y el aprendizaje de lenguas extranjeras, por ejemplo. Con base en los resultados de la investigación, la autora concluye que los métodos simples combinados presentan una posibilidad de introducción temprana en la programación en las escuelas secundarias.

Bell y otros [19], integran ALICE en un proyecto de ciencias de la computación fuera de línea, el cual provee maneras de exponer a los estudiantes a las ideas de las ciencias computacionales sin tener que usar computadoras. En el proyecto, utilizan actividades, juegos, trucos de magia y concursos para mostrar a los estudiantes el tipo de pensamiento que se espera de un informático; así mismo, incluyen adaptaciones de las actividades cinestésicas en mundos virtuales, integran herramientas como ALICE, e incluyen videos para ayudar a los docentes a comprender el uso del material.

Kelleher y Pausch [86], usan ALICE para motivar a estudiantes de género femenino. Plantean que se debe crear planes de estudio basados en juegos donde ambos géneros (masculino y femenino) sean capaces de motivarse; a los estudiantes de ambos sexos; y que a menos que se tenga cuidado en los diseños de los planes de estudio de juegos, se puede ampliar la diversidad en informática.

Ko y Myers [89], reportan experiencias utilizando un esquema para estudiar ALICE, afirmando que el mismo había inspirado directamente el diseño de nuevas herramientas de programación e interfaces. Estos autores, proponen un marco y una metodología que se centra específicamente en los errores apoyando la descripción y determinar las causas de los errores de software en términos de cadenas de averías cognitivas. Entre los resultados, se destaca la inclusión de una interfaz de depuración, demostrando que reducen el tiempo de depuración en un factor de 8 y la ayuda a los programadores obteniendo un 40 % mas a través de sus tareas. Adicionalmente, discuten las implicaciones del marco y la metodología para la programación del diseño de sistemas, ingeniería de software y la psicología de la programación.

Otros autores como Mullins y otros [110], reportan el uso de ALICE como lenguaje para el curso introductorio de programación.

Entre las tesis y trabajos de grado fundamentados en la aplicación del sistema de programación ALICE, se destacan Dwarika [50], con su trabajo *"The use of ALICE, a visual environment for teaching and learning object-oriented programming"*, Daly [44] con su disertación doctoral *Ïnfluence of Alice 3: Reducing the Hurdles to Success in a CS1 Programming Course"*, Sharma [145], con su trabajo doctoral *"Library automation software packages used in academic libraries of Nepal: a comparative study"*, cuyos resultados evidencian que ALICE es el mejor, colocándolo primero en la evaluación realizada.

4.5. Resumen

Los resultados de las diferentes investigaciones revisadas han evidenciado que utilizar la herramienta ALICE ayuda a que los estudiantes comprendan conceptos básicos con el paradigma orientado a objetos, así como el uso de estructura de control. De igual manera, facilita la articulación vertical entre asignaturas de las carreras. Entre las fortalezas de ALICE reportadas, se pueden destacar, la solución del problema de la sintaxis, el hecho de permitir el avance durante el desarrollo del programa, la reducción del tiempo de desarrollo, y particularmente, el abordaje de la motivación para aprender a programar, considerada como una de las razones para alejar a los estudiantes de los cursos de programación. Por otro lado, es importante la abundante cantidad de literatura donde se documenta las buenas prácticas del uso de la herramienta ALICE.

Parte III

Contribuciones

Capítulo 5

Motivación

Si quieres dominar algo, enséñalo. Cuanto mas enseñas, mejor aprendes. La enseñanza es un herramienta poderosa para el aprendizaje.

Richard Feynman

En este capítulo, se presentan algunos motivos que impulsan el uso de lenguajes de programación educativos y la eficacia de la aplicación de ALICE en el proceso de aprendizaje de la programación en estudiantes de nivel superior. En la Sección §5.1 se describen los motivos de realizar este estudio; más tarde en la Sección §5.2 se analizan los principales problemas del presente trabajo. En la Sección §5.3 hacemos un resumen de las soluciones propuestas; luego en la Sección §5.4 se discute sobre algunos trabajos relacionados. Finalmente, resumimos el Capítulo en la Sección §5.5.

5.1. Introducción

Las dificultades que presentan los estudiantes, en un nivel universitario, para desarrollar el pensamiento computacional y aprender programación, ha despertado el interés del presente estudio. Por tanto, esta tesis se centra principalmente en:

1. Determinar los LPEs que podrían utilizarse herramienta en el aprendizaje de la programación.
2. Evaluar el impacto de la aplicación de un LPE en el nivel universitario.

Los entornos de programación visual basados en bloques, usados hoy en día por millones de estudiantes [144], son una buena estrategia para la introducción, inmersión y participación a los nuevos programadores en el mundo de la programación informática [27, 180]. Aunque este enfoque no es una innovación reciente, se ha generalizado en los últimos años con el surgimiento de una nueva generación de herramientas diseñadas bajo un entorno visual [171].

Los entornos de programación visual basados en bloques son una de las opciones más usadas y de mayor efectividad para aprender programación y mejorar efectivamente las habilidades el pensamiento computacional en los estudiantes [150]; dentro de este grupo se incluyen entornos como Scratch y ALICE, cuyas evidencias reportadas la posicionan como alternativa amigable para los nuevos programadores y para cultivar habilidades de pensamiento computacional [114, 150].

El objetivo de este capítulo es motivar sobre el uso de los lenguajes de programación educativo en un nivel universitario, para que los estudiantes de las carreras de informática desarrollen habilidades de programación y de pensamiento computacional.

5.2. Problemas

Uno de los métodos para desarrollar la capacidad de pensamiento computacional desde una edad temprana es la programación [28]. Sin embargo, la falta de pensamiento lógico, creativo y crítico entre los estudiantes que se inician en la programación suelen ocasionar dificultades y fracasos en aspectos de resolución de problemas [81]. Por otra parte, la interfaz de usuario de los lenguajes tradicionales requiere que los estudiantes aprendan muchos comandos, lo que hace que la mayoría se sientan aburridos y sin interés en el aprendizaje de la programación.

A nivel general, el desarrollo de nuevas tecnologías requiere de más profesio-

nales en el área de desarrollo de software. Esta área ha superado una rápida expansión y con una tendencia creciente, por lo que requiere que cada programador tenga que aprender constantemente y dominar las nuevas tecnologías [105]. Esta situación representa un desafío para la sociedad moderna al preparar a las nuevas generaciones de desarrolladores de software, ya que requiere personas con conocimientos en algoritmos y programación de computadoras [26].

La programación informática en general es algo difícil de aprender y de enseñar, según algunos estudios realizados con estudiantes de grado [45], [61], [83], en los que la programación forma parte de su formación. Los problemas se deben sobre todo a que: I) hay pocos estudios que investiguen la pedagogía de los lenguajes de programación, II) se necesita una comprensión más profunda de cómo aprenden los estudiantes en general, y; III) a menudo pierden la atención y la motivación por el aprendizaje.

En el presente trabajo de investigación se abordan los siguientes problemas:

1. Se necesita información más profunda sobre la forma de reportar la efectividad después de la aplicación del un LPE.
2. No se cuenta con suficiente información sobre los objetivos pedagógicos que se logran con el uso de LPEs.
3. Hay pocos estudios que reporten la evidencia empírica del uso los LPEs, específicamente de ALICE.

Como resultado, cuando los esfuerzos por aprender a programar fracasan, la culpa puede no estar en la tecnología, sino en la falta de una estrategia o metodología adecuadas.

5.3. Análisis de soluciones propuestas

El objetivo de esta sección es argumentar que ninguno de los enfoques presentados en los capítulos §3 y §4, abordan los problemas antes mencionados:

5.3.1. Efectividad del uso de LPE

No se ha encontrado ninguna trabajo que reporte la efectividad tras la aplicación de los LPEs en el proceso de aprendizaje de la programación. Es necesario considerar las alternativas con las cuales se puede evaluar la efectividad y poder comprobar el efecto del LPE en el aprendizaje y desarrollo del pensamiento computacional o en su defecto habilidades como la resolución de problemas.

5.3.2. Objetivos pedagógicos del uso de LPE

Con respecto a los objetivos pedagógicos, en la literatura se identificó que en el trabajo realizado por Salleh y otros [136] se reportó los objetivos pedagógicos planteados en la literatura, no se detectaron mas trabajos que hayan reportado sobre este tema. Aunque hay poca información sobre los objetivos pedagógicos, se debería considerar ofrecer una clasificación de los objetivos teniendo en cuenta las habilidades que se pueden adquirir al aprender a programar.

5.3.3. Evidencia empírica del uso de ALICE

Hay algunos estudios en la literatura que reportan información sobre la aplicación del software ALICE y su uso en el proceso de aprendizaje de la programación [6, 41, 49, 175] en distintos niveles educativos. Los estudios analizados en su mayoría se han realizado en países de la comunidad WEIRD (Occidental, Educado, Industrializado, Rico y Democrático), como son Estados Unidos y Portugal. Sin embargo, es evidente la necesidad orientar el estudio a comunidades non-WIERD las cuales se comportan de manera diferente por su característica de no industrializados ni ricos.

5.4. Discusión

Algunos estudios previos presentan resultados de revisiones sistemáticas de la literatura al abordar información sobre la aplicación de uno o varios LPEs en el proceso de aprendizaje de la programación.

En la Tabla §5.1 se presentan otros estudios de autores que aplicaron el método de RSL, [42, 99, 101, 109, 114, 136, 155] que fueron publicados en los últimos años, el período utilizado en las revisiones por los autores varía según el contexto del estudio, también se muestra una comparación con cuatro aspectos considerados en la presente investigación y el trabajo realizado por los autores.

El análisis comparativo se realizó con respecto a cuatro aspectos que conforman el ámbito de nuestra investigación, que corresponden a: la evidencia empírica del método elegido en los trabajos analizados, el contexto educativo en el cual se realizan las investigaciones, la eficacia de la aplicación del LPE, y los objetivos pedagógicos que se buscaban alcanzar en los trabajos examinados.

En cuanto al método empírico seleccionado para la investigación, la tabla comparativa muestra que los autores [42], [99], [101] y [109] han reportado el método utilizado en sus publicaciones, es necesario enfatizar que la evaluación de

Trabajos de revisión	Evidencia empírica	Contexto educativo	Efectividad del LPE	Objetivos pedagógicos
Costa *et al.* [42]	✓	✓	-	-
Lye *et al.* [99]	✓	-	-	-
Medeiros *et al.* [101]	✓	-	-	-
Moreno *et al.* [109]	✓	✓	-	-
Salleh *et al.* [136]	-	✓	-	✓
Ting *et al.* [155]	-	✓	-	-
Presente tesis	✓	✓	✓	✓

Tabla 5.1: Comparación de estudios relacionados

la calidad del método (también considerada en esta investigación) sólo se presenta en el presente trabajo, el cual puede ser tomado como referencia para que los autores de próximas publicaciones consideren la inclusión de este tema en sus futuras investigaciones.

En cuanto al contexto de la investigación, los autores [99] y [101] no informan sobre el contexto educativo que son los aspectos relacionados con el país y la etapa educativa donde se aplicaron sus estudios.

Por último, en relación con el análisis de los objetivos pedagógicos alcanzados, se puede observar que sólo [136] los reportó. Cabe señalar que la eficacia sólo se presenta en el presente trabajo, que puede ser tomado como referencia para que los autores de futuras publicaciones puedan considerar la inclusión de este tema en sus investigaciones.

Considerando que no existe suficiente información sobre la aplicación de un LPE en el contexto universitario, referente a (I) evidencia empírica de los métodos utilizado, (II) el contexto educativo; (III) la efectividad de la aplicación del LPE, o; (IV) los objetivos pedagógicos que se desean obtener, es necesario obtener más información sobre los aspectos mencionados que permita establecer los lenguajes educativos más utilizados, y a su vez identificar el nivel educativo apropiado para la aplicación del mismo.

Por lo tanto, la presente investigación tiene como objetivo recopilar la mayor cantidad de información sobre la aplicación de los LPEs en los diferentes niveles educativos y determinar su aficacia en el proceso de aprendizaje de la programación. Mediante una RSL, analizar las ventajas o desventajas de utilizar diferentes métodos o herramientas; y, con la realización de un experimento poder evaluar la eficacia de ALICE como herramienta para el aprendizaje de

la programación. Se espera que este estudio ayude a determinar alguna estrategia para incrementar el interés de los estudiantes y su rendimiento en la asignatura de programación.

5.5. Resumen

El interés de la comunidad académica sobre el uso de los recursos para facilitar el aprendizaje de la programación de los estudiantes principiantes va en incremento, algunos lenguajes de programación basados en comandos y bloques facilitan el aprendizaje y permiten desarrollar destrezas para el aprendizaje de programación. Se evidencian resultados de las buenas prácticas en los primeros años de las carreras en nivel universitario. La mayoría de las estudios donde de experimentaron con ALICE en un contexto educativo, evidencian la efectividad de esta plataforma como herramienta de apoyo en el proceso de aprendizaje de la programación.

Capítulo 6

Evidencias empíricas en el uso de lenguajes de programación educativos

El mejor regalo que un educador puede dar es conseguir que alguien se convierta en autodidacta.

Randy Pausch

En este capítulo, se presentan las evidencias empíricas de los diferentes estudios encontrados y que fueron analizaron referente al proceso de aprendizaje de la programación. Las evidencias de la revisión bibliográfica incluyen un resumen de los métodos elegidos para mejorar el proceso de aprendizaje de la programación, los lenguajes de programación aplicados en cada trabajo y los objetivos alcanzados en cada estudio. En la Sección §6.1 se especifican los problemas que presentan los estudiantes en el proceso de aprendizaje de la programación; más adelante en la Sección §6.2 se analiza el procedimiento utilizado en el presente estudio. Luego, en la Sección §6.3 se presentan los resultados de la revisión de la literatura realizada. Finalmente, resumimos este capítulo en la Sección §6.4[†1].

[†1]Parte de este capítulo está publicado en IEEE Transactions on Education [160]. Se tiene el acuerdo de todos los autores para incluir el texto correspondiente como parte de este trabajo.

6.1. Introducción

Las herramientas educativas utilizadas para aprender habilidades de las TIC, han despertado la atención de los investigadores en el ámbito educativo. Esto se debe a que muchos de los estudiantes creen que la programación es una asignatura difícil de aprender, así como la falta de motivación e interés que presentan para adquirir habilidades de programación.

En este capítulo se reporta los resultados de la revisión de la literatura realizada con el uso de los LPEs, utilizados como instrumento de aprendizaje, se informa que los estudiantes mejoraron significativamente sus destrezas de programación y desarrollo del pensamiento computacional, a través de entornos visuales basados en bloques y lenguajes de programación textuales, que son dos de las cinco categorías para desarrollar el pensamiento computacional mencionadas por [108]. El uso de estas dos categorías podría indicar que aún no hay consenso entre cuál es la más adecuada para abordar el aprendizaje de la programación.

Así mismo, al evaluar la literatura, se observó que alrededor del 60 % de los estudios buscan la mejora del pensamiento computacional mediante la programación. Se detectó que solo alrededor del 14 % de los estudios con evidencia empírica fueron realizados en etapas educativas tempranas, lo que indicaría que se necesita mas evidencia empírica en estos niveles educativos. De la evidencia empírica reportada, se descubrió que se adoptaron principalmente dos estrategias: casos de estudio y experimentos controlados. Sin embargo, el nivel de madurez de estos estudios es relativamente bajo, lo que dificulta replicar por parte de los investigadores. No obstante, la gran mayoría de los trabajos informan que consiguen el objetivo que persiguen con sus investigaciones realizadas.

Al analizar los trabajos existentes y responder a las preguntas de investigación, se identifican ciertas oportunidades de investigación, como la necesidad de:

(1) Detectar en qué áreas temáticas del programa de estudio se utiliza el pensamiento computacional.

(2) Determinar en qué medida se utilizan las diferentes herramientas existentes para el progreso del aprendizaje.

(3) Utilizar diferentes alternativas de lenguajes de programación para comparar los resultados entre ellas.

(4) Disponer de más evidencias empíricas de mejor calidad.

(5) Desarrollar mas estudios y evidencias empíricas en países pertenecientes a non-WEIRD.

(6) El desarrollo de estudios sobre el uso de lenguajes de programación textual en etapas educativas tempranas.

A través de la revisión de literatura que se realizó se pretende abordar aspectos relacionados con el uso de los lenguajes de programación educativos y su incidencia en el proceso de enseñanza de la programación.

6.2. Procedimiento

Para establecer los LPEs que se utilizan como herramienta en el proceso de aprendizaje de la programación, se realizó una revisión sistemática de literatura sobre evidencias empíricas relativas al uso de estos lenguajes con fines educativos. La revisión analiza diferentes métodos y herramientas en diferentes niveles educativos y con diferentes objetivos.

La presente investigación sigue las directrices de Kitchenham y otros [88] para el procedimiento realizado en la RSL. La Figura §6.1 muestra las principales fases para llevar a cabo este proceso, que se describen a continuación:

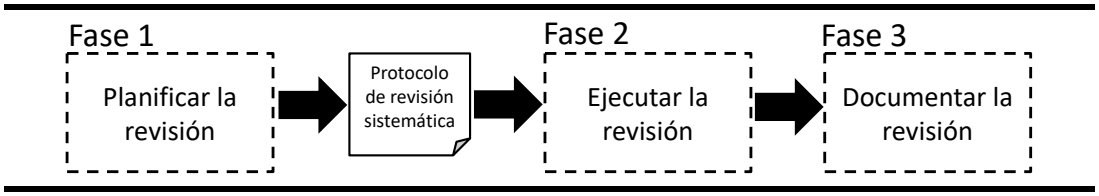

Figura 6.1: Proceso de revisión sistemática de literatura [88]

A continuación se detalla el proceso que guían esta RSL:

6.2.1. Fase 1: Planificar la RSL

En esta primera fase se diseña la forma se que se realiza el estudio y la documentación que se genere a través del protocolo de revisión. Implica la ejecución de dos actividades del proceso, como se presenta en la Figura §6.2. A continuación, se detallan cada uno de estas actividades.

6.2.1.1. Especificar las preguntas de investigación

El objetivo de esta revisión bibliográfica es responder las siguientes preguntas de investigación:

1) *RQ1: ¿Qué evidencia empírica existe sobre el uso de los Lenguajes de programación educativos?*

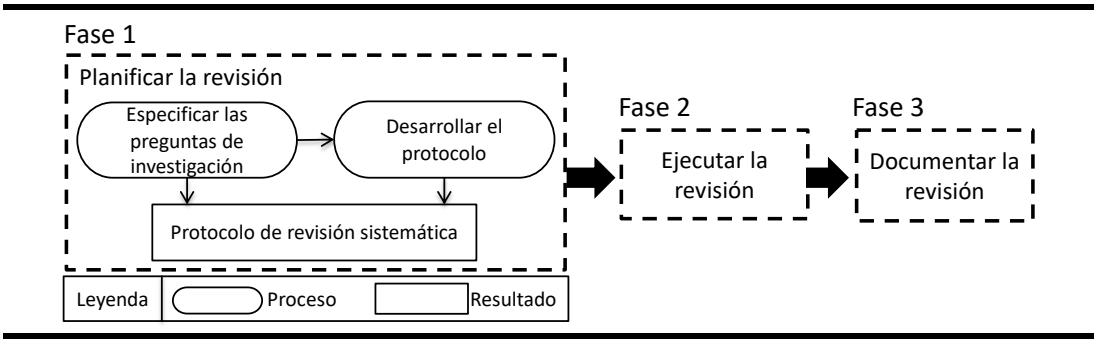

Figura 6.2: Fases de la planificación [88]

2) *RQ2: ¿En qué contexto se realiza la investigación?*

3) *RQ3: ¿Como se reporta en la literatura la eficacia después de la aplicación de un Lenguaje de programación educativo?*

4) *RQ4: ¿Qué objetivos pedagógicos se logran con el uso de Lenguajes de programación educativos?*

6.2.1.2. Desarrollar el protocolo

En esta actividad se elabora el documento del protocolo que describe todos los detalles de cómo se realizará el proceso de la revisión de la literatura. El protocolo presentado en la guía de Kitchenham y otros [88] abarca elementos como los antecedentes, las preguntas de investigación, las estrategias de búsqueda, la extracción de datos, las bases datos y los criterios de inclusión / exclusión utilizados para el presente estudio. El diseño de cada fase del protocolo se explica en la siguiente Sección §6.2.2.

6.2.2. Fase 2: Ejecutar la RSL

En esta fase se aplica el plan establecido en el protocolo de la revisión sistemática. En la Figura §6.3 se presentan las actividades a realizar en esta fase. A continuación se detalla la realización de cada una de estas actividades.

6.2.2.1. Identificar la investigación

En esta actividad de la revisión se determinan las bases de datos a consultar. Concretamente, las bases de datos que se utilizaron en la presente investiga-

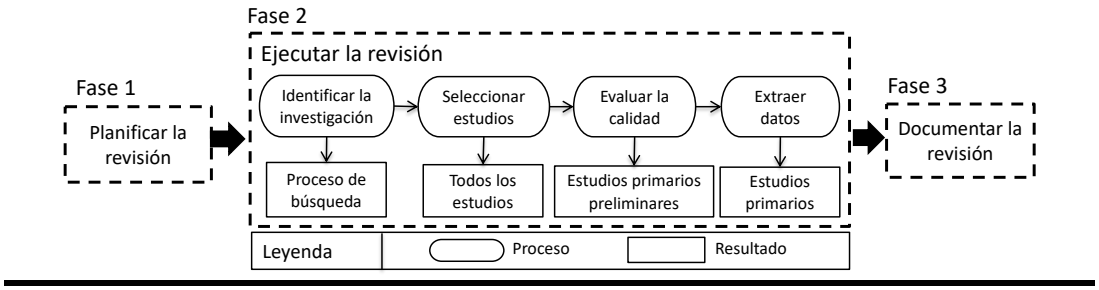

Figura 6.3: Fases de la ejecución [88]

ción son Scopus[†2], ACM[†3], IEEE[†4], Sicence Direct[†5] y Wiley[†6], basado en que son fuentes muy consultadas y que contienen la mayor cantidad de información. Hay que tener en cuenta que se han utilizado las mismas bases de datos que en el estudio de RSL realizado por [109] que son fuentes de información bibliográficas habituales en la comunidad investigadora.

La ejecución de los criterios de búsqueda en las bases de datos se realizó en noviembre de 2018, la cadena de búsqueda fue la siguiente:

```
("Learning programming.ºR "Learning coding.ºR "Learning
to program.ºR "Learning to code.ºR "Learning of computer
programming") AND (.Educational software.ºR .Educational
technology.ºR .Educational programming language.ºR "Visual
programming environment.ºR "Block-based languages.ºR
"Block-based coding")
```

En la Tabla §8.1 del Apéndice A, se presenta la cadena de búsqueda utilizadas para cada base de datos. Al ejecutar la consulta, se recuperaron 899 artículos de todas las bases de datos.

6.2.2.2. Seleccionar publicaciones

Al ejecutar las cadenas de búsqueda en cada base de datos, se siguieron los pasos presentados en la Figura §6.4, y seleccionar los trabajos relevantes para el presente estudio.

Después de ejecutar la consulta de las bases de datos, los estudios recupera-

[†2] http://www.scopus.com
[†3] https://dl.acm.org/
[†4] https://ieeexplore.ieee.org/Xplore/home.jspg
[†5] https://www.sciencedirect.com/
[†6] https://onlinelibrary.wiley.com/

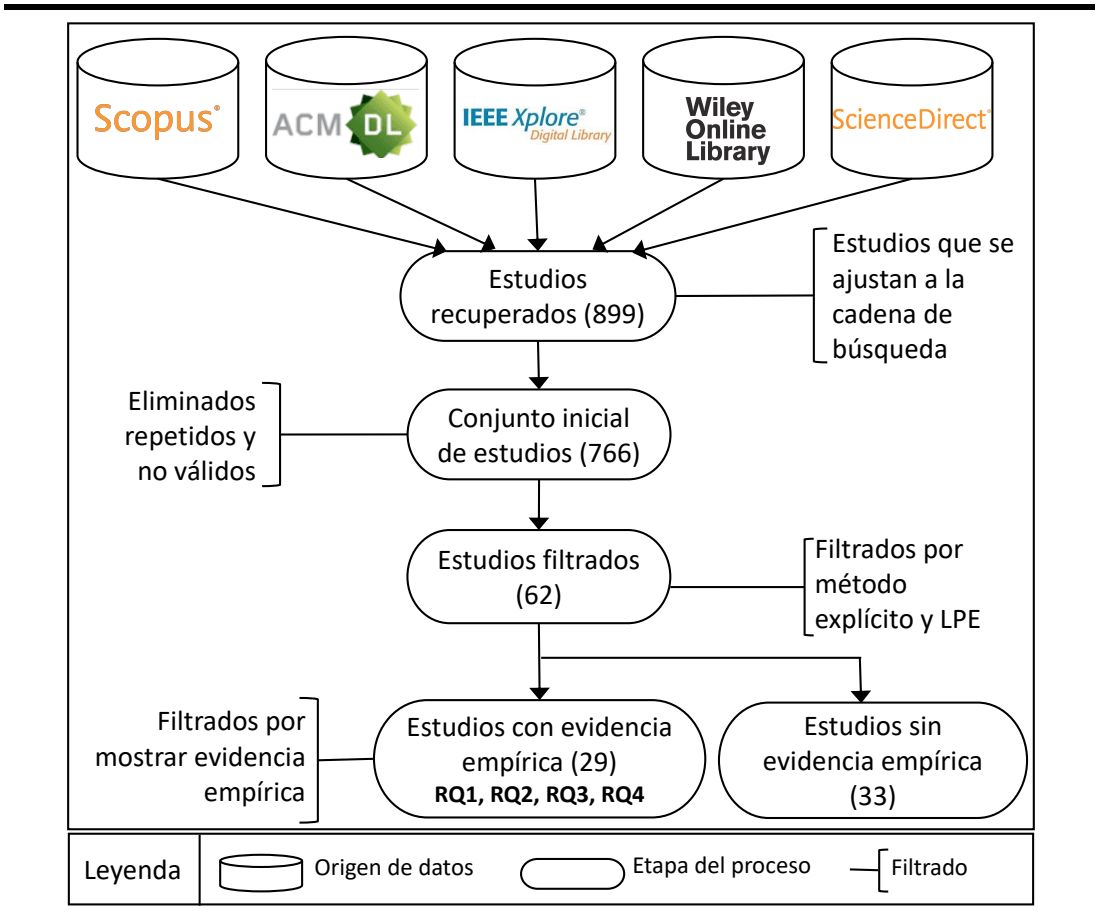

Figura 6.4: Proceso de búsqueda de publicaciones

dos se sometieron a un análisis en profundidad, considerando los siguientes criterios de inclusión y exclusión:

Criterios de inclusión:

1) Trabajos publicados desde enero de 2007, considerando que el desarrollo de Scratch comenzó en 2004 y fue finalmente lanzado en el 2007.

2) El tipo de documento debe ser un *journal article* o *conference paper*.

3) Trabajos escritos en inglés.

4) Trabajos en las siguientes disciplinas: ciencias de la computación, ciencias sociales, educación y computación.

5) Trabajos que indiquen explícitamente el método y el Lenguaje de programación educativo utilizado en el proceso de aprendizaje de la programación.

Base de datos	#Artículos recuperados (1)	#Artículos eliminados (2)	#Artículos filtrados (3)	#Artículos con evidencia (4)
Scopus	39	31	19	9
ACM	36	25	14	5
IEEE	12	8	2	1
Science Direct	438	390	22	11
Wiley	374	312	5	3
Total	899	766	62	29

Tabla 6.1: Número de artículos identificados en la investigación

6) Trabajos que demuestren evidencia empírica de sus resultados.

Criterios de exclusión:

1) Trabajos cuyos estudios no se aplicaron en el aprendizaje de la programación.

2) Trabajos duplicados.

3) Trabajos no válidos, es decir, que no se centren en la aplicación de un Lenguaje de programación educativo para el aprendizaje de la programación.

La Tabla §6.1 presenta el número de trabajos identificados en la revisión de la literatura, aplicando los criterios de inclusión y exclusión detallados en esta sección.

La descripción de las columnas es la siguiente: (1) el número de estudios identificados al buscar en cada base de datos aplicando la cadena de búsqueda de la sección §6.2.2; (2) el número de trabajos eliminados por ser duplicados o inválidos para la investigación; (3) el número de trabajos filtrados que declaran explícitamente el método que se utilizó en el proceso de aprendizaje y el lenguaje de programación educativo, y; (4) el número de estudios filtrados que muestran evidencia empírica del procedimiento de aprendizaje.

6.2.2.3. Evaluar la calidad

En primer lugar, cabe señalar que los criterios de inclusión y exclusión aplicados, detallados anteriormente, se consideran un primer indicador de calidad. Se utiliza para evaluar las contribuciones seleccionando el mayor número de

estudios y asegurándose de que informan de investigaciones que proporcionan resultados empíricos de los métodos aplicados.

En segundo lugar, y para las demás preguntas de investigación consideradas, la evaluación de la calidad se realiza posteriormente y no como una forma de descartar artículos para el análisis. Esto se debe a que la preguntas de investigación RQ1 (concretamente en RQ1.2) se trata de la evaluación de la calidad de los artículos revisados; además es por ello que el resultado de dicha evaluación no se utiliza para excluir los artículos del presente estudio ya que forma parte del análisis presentado en la Sección §6.3.

6.2.2.4. Extraer los datos

Una vez concluidas las fases descriitas anteriormente, se clasifican los datos obtenidos en la última etapa de la fase dos y se siguen los siguientes pasos:

1) Se crea un repositorio para almacenar todos los artículos que se van a clasificar. Concretamente, se utiliza la base de datos bibtex[†7]. Posteriormente, se gestiona con Mendeley[†8].

   La base de datos registra los siguientes atributos para cada artículo: a) el tipo de documento; b) el título; c) los autores; d) el año de publicación; e) un resumen; y, f) el enlace para acceder al documento completo.

2) Aplicando los criterios de selección, se obtienen como resultado 62 artículos. A continuación, se crea una base de datos en Excel añadiendo la siguiente información: a) el lugar donde se realizó la investigación (país e institución educativa); b) el método aplicado; c) el software utilizado; d) el objetivo; y, e) los resultados obtenidos según las preguntas de investigación.

3) De estos 62 estudios, se separan los que presentan resultados empíricos, dando lugar a 29 artículos. Estos últimos artículos forman el segundo grupo de trabajos, y para ellos se incluye la siguiente información: a) el contexto educativo en el que se aplicó el método; y, b) los detalles del método utilizado según las directrices propuestas por Wohlin y otros [177].

6.2.3. Fase 3: Documentar la RSL

En esta última fase de la RSL, que se presenta en la Figura §6.5, se realiza la clasificación y documentación de la información recuperada de los artículos

[†7] http://www.bibtex.org/
[†8] https://www.mendeley.com/

considerados en el presente estudio (Véase la Sección §6.3). Para ello, se responde a las preguntas de investigación incluidas en la Sección §6.2.1.1 y se exponen los resultados del análisis de los 29 artículos que muestran evidencia empírica.

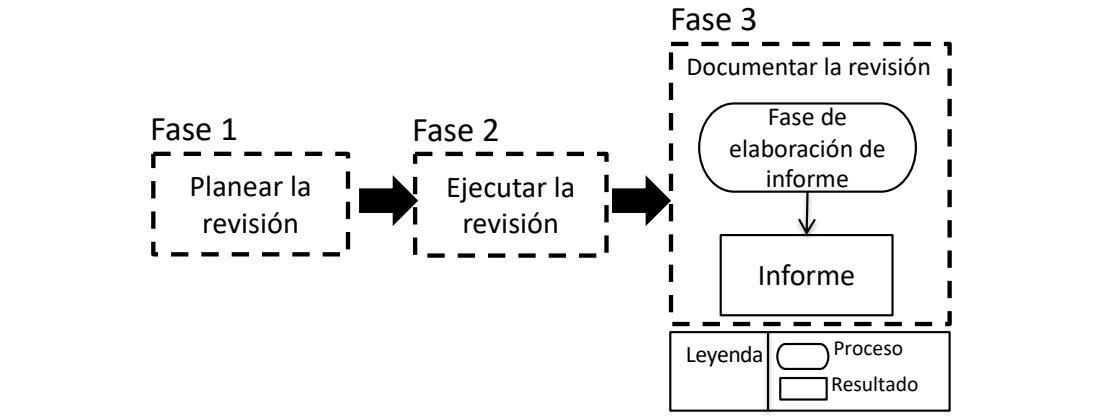

Figura 6.5: Fase de documentación [88]

6.3. Resultados

En la presente sección se muestran los resultados de la revisión de la literatura realizada sobre los trabajos que presentan evidencia empírica sobre el método de investigación utilizado.

6.3.1. Evidencia empírica en el uso de los LPE

En esta sección se da respuesta a la pregunta de investigación:

- *RQ1: ¿Qué evidencia empírica existe sobre el uso de los lenguajes de programación educativos?*

En el contexto de esta pregunta se analizaron dos situaciones: 1) los métodos empíricos descritos en la literatura; y, 2) los criterios que establecidos para determinar la calidad del método y comunicar los resultados. Lo que motiva las siguientes subpreguntas:

1) RQ1.1 ¿Cuáles son los métodos empíricos utilizados?
2) RQ1.2 ¿Qué criterios se han establecido para determinar la calidad del método e informar de los resultados?

A continuación se detallan los resultados:

Método	Cita	#Artículos
Estudios de caso	[35, 82, 90, 103, 112, 113, 116, 117, 124, 135, 151, 156, 169, 175, 184]	15
Experimento	[9, 20, 55, 58, 94, 102, 123, 129, 167, 185]	10
Otros	[20, 57, 72, 74, 82, 132, 151]	7

Tabla 6.2: Clasificación de los artículos basados en el método de investigación

6.3.1.1. Métodos empíricos utilizados

En esta sección presentan los métodos empíricos descritos en la literatura y que han sido utilizados para el proceso de aprendizaje de la programación. En consecuencia, para determinar los métodos, se utiliza el grupo de 29 trabajos que presentaron evidencias empíricas. Algunos estudios utilizan mas de un método. La clasificación se muestra en la Tabla §6.2

a) **Estudio de casos:** los artículos de esta categoría utilizan el estudio de caso como método de investigación empírica, por ejemplo, Runeson y otros [133], investigan un caso de un evento moderno dentro de su contexto de la vida real.

b) **Experimento:** los artículos de esta categoría utilizan el experimento como método de investigación empírica, que se lleva a cabo principalmente en un entorno de laboratorio, lo que proporciona un alto nivel de control, por ejemplo, Wohlin y otros [177], que tiene como objetivo manipular una o mas variables para realizar un análisis estadístico.

c) **Otros:** los artículos de esta categoría utilizan otros métodos de investigación, como cursos o talleres, para aplicar un LPE en el proceso de aprendizaje de la programación.

Se observa que el método más utilizado en la literatura es el estudio de casos con 15 artículos que representan el 47 % del total, al que le siguen los experimentos con diez artículos que representan el 31 % del total. Además, los autores de siete trabajos eligieron otros métodos, como el taller y los cursos, que representan el 22 % del total. De estos resultados se deduce que los investigadores consideran que los estudios de casos son mas fáciles de planificar y mas realistas, mientras que los experimentos tratan de aislar la situación estudiada del mundo real y son mas tediosos.

6.3.1.2. Criterios para determinar la calidad del método

En esta sección se analizan los criterios que se tienen en cuenta a la hora de comunicar los resultados. Para evaluar los métodos, se consideraron las estructuras de los informes del experimento y del estudio de casos propuesta por Wohlin y otros [177]. Cada criterio era una variable binaria con un valor de 1 si se reportaba la información correspondiente y 0 en caso contrario.

Los siguientes aspectos corresponden al estudio de casos:

a) **Introducción:** en relación con este tema, la definición del planteamiento del problema, los objetivos de la investigación y el contexto en el que se realizó el estudio se consideraron parte de la introducción.

b) **Diseño del estudio de casos:** Incluye la definición de las preguntas de investigación, la selección de casos y sujetos, el procedimiento de recogida de datos, el procedimiento de análisis y/o el procedimiento de evaluación de la validez.

c) **Resultados:** Conlleva la descripción del caso y del sujeto que abarca la ejecución, los aspectos de análisis e interpretación, las secciones que pueden estructurarse, y/o la evaluación de la validez.

d) **Conclusiones y trabajo futuro:** Se componen de un resumen de los resultados, relaciones con la evidencia existente, impacto/implicaciones, limitaciones y/o trabajo futuro.

La Tabla §6.3 presenta los diferentes aspectos e identifica 15 estudios en los que se aplicó el método de *estudio de casos*. En el 100 % de estos estudios, se describió toda la información de la *Introducción*. En el 55 % de los estudios se informó sobre el *Diseño del estudio de caso*; solo el 27 % de los estudios representa información sobre los *Resultados* de la investigación. Aproximadamente el 38 % de los trabajos presentan información sobre las *Conclusiones y trabajos futuros*. Chao [35] y Navarrete [112] son los que han informado de los estudios mas completos.

Los aspectos considerados para el experimento se muestran a continuación:

a) **Motivación:** este ítem considera si se definió el estado del problema, los objetivos de la investigación y el contexto en el que se realizó el estudio.

b) **Diseño experimental:** incluye las hipótesis, el diseño, los sujetos, los objetos, la instrumentación, el procedimiento de recolección de datos, el procedimiento de análisis y la evaluación de la validez.

c) **Ejecución:** comprende la muestra, la preparación, la recogida de datos realizada y el procedimiento de comprobación de la validez.

#	Tópico	Chao [35]	Kafai *et al.* [82]	Kordaki [90]	Merkouris *et al.* [103]	Navarrete [112]	Nikou *et al.*[113]	Ortega *et al.*[116]	Papadakis *et al.*[117]	Reardon *et al.*[124]	Saez-López *et al.*[135]	Tanrikulu *et al.*[151]	Tuparov *et al.*[156]	Webb [169]	Woei *et al.*[175]	Yoon *et al.*[184]
	Introduction	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	Problem statement	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	Research objectives	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	Context	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Case study design	4	4	4	3	4	3	2	2	3	3	1	1	1	4	2
4	Research questions	1	1	1		1				1					1	
5	Case and subject selection	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	Data collection procedure	1	1	1	1	1	1		1	1	1				1	
7	Analysis procedure	1	1	1	1	1	1	1			1				1	1
8	Validity procedure															
	Results	3	0	1	1	1	0	1	0	2	2	0	0	0	1	0
9	Case and subject descriptions	1			1			1								
10	Subsection	1		1		1				1	1				1	
11	Evaluation of validity	1								1	1					
	Conclusions and future work	2	2	2	2	3	2	2	0	1	1	1	2	1	1	1
12	Summary of findings	1	1	1	1	1	1	1		1	1	1	1	1	1	1
13	Relation to existing evidence															
14	Impact/implications					1										
15	Future work	1	1	1	1	1	1	1					1			
	Total	12	9	10	9	11	8	8	5	9	9	5	6	5	9	6

Tabla 6.3: Criterios para determinar la calidad de los estudios de caso

d) **Análisis:** contiene la estadística descriptiva, la exploración del conjunto de datos y la prueba de hipótesis.

e) **Interpretación:** incluye la evaluación de los resultados, las limitaciones del estudio, las inferencias y las lecciones aprendidas.

f) **Conclusiones y trabajo futuro:** por último, las conclusiones están relacionadas con las pruebas existentes, los impactos las limitaciones y el trabajo futuro.

En la Tabla §6.4 se identifican 10 autores que realizaron *experimentos* en sus investigaciones. De los criterios establecidos para evaluar la calidad de los trabajos, los aspectos en los que se ha reportado mas información corresponden a la *Motivación* con el 100 % de los trabajos. A continuación, el 49 % de los trabajos informaron sobre el *Diseño experimental*, el 47 % sobre el *Análisis*, el

43 % sobre la *Interpretación* y, finalmente, el 38 % presentó información sobre la *Ejecución* y las *Conclusiones y trabajos futuros*. En cuanto a los autores, Feng y Chen [58] son los que mas información han aportado sobre el estudio realizado, seguidos de Lee y Ko [94] y Wang y otros [167]. Los demás autores no han aportado suficiente información.

En general, se observa que no hay consistencia en lo reportado y que falta madurez en el tema. Además, se observa en falta un informe completo que fomente la disciplina en la redacción de resultados. Es necesario conocer los detalles para poder replicar los respectivos trabajos en entornos similares y poder detectar si el lector puede determinar con claridad y facilidad lo que se estudió y cómo se obtuvieron los resultados presentados.

6.3.2. Contexto educativo de los estudios empíricos

En esta sección se da respuesta a la pregunta de investigación

- *RQ2: ¿En que contexto se realiza la investigación?*

Para responder a esta pregunta, se analiza el contexto educativo en el que se han realizado las investigaciones empíricas, incluyendo el continente y el nivel educativo de los participantes en el estudio, y sólo se han considerado los 29 artículos que presentan evidencias empíricas.

En la Figura §6.5, se muestran los países en los cuales los autores han realizado sus respectivos estudios. En general, se observa que el mayor número de trabajos han sido publicados por organizaciones europeas, sumando 13 artículos. Le siguen América del Norte, con cinco artículos y Asia, con cuatro artículos. Sin embargo, los autores de siete artículos no han especificado dónde se realizó la investigación.

Al parecer este tipo de estudios se realizan principalmente en Europa porque los investigadores cuentan con mas información y recursos. Sin embargo, sería importante analizar los factores que influyen en la investigación en un sitio u otro y que oportunidades de investigación se originan.

En la Figura §6.6, se muestran los países en los que los autores han realizado sus respectivos estudios, seguido de las lista de instituciones y el país al que pertenecen.

Además, se han tenido en cuenta las etapas educativas y se han establecido según la organización responsable de la educación y la formación profesional en España, para clasificar los 29 artículos.

La Tabla §6.6 muestra las etapas educativas, establecidas según el organismo

#	Tópico	Alshaigy *et al.*[9]	BenAri *et al.* [20]	Esteves *et al.*[55]	Feng *et al.*[58]	Lee *et al.*[94]	Merkouris *et al.*[102]	Rahman [123]	Rogozhkina *et al.*[129]	Wang *et al.*[167]	Yukselturk *et al.*[185]
	Motivation	3	3	3	3	3	3	3	3	3	3
1	Problem statement	1	1	1	1	1	1	1	1	1	1
2	Research objectives	1	1	1	1	1	1	1	1	1	1
3	Context	1	1	1	1	1	1	1	1	1	1
	Experimental design	4	3	4	7	5	3	4	3	4	2
4	Hypothesis	1			1	1				1	
5	Design	1		1	1		1	1	1		1
6	Subjects	1	1		1	1	1	1	1	1	1
7	Objects	1	1	1	1	1		1	1	1	
8	Instrumentation		1	1	1	1				1	
9	Data collection procedure				1	1	1	1			
10	Analysis procedure			1	1						
11	Evaluation of validity										
	Execution	2	2	1	3	1	1	1	2	1	1
12	Sample	1	1		1	1	1	1	1	1	1
13	Preparation	1	1		1				1		
14	Data collection performed			1	1						
15	Validity procedure										
	Analysis	0	1	0	3	2	2	1	2	1	2
16	Descriptive statistics		1		1	1	1	1	1	1	1
17	Data set reduction				1	1		1		1	
18	Hypothesis testing				1		1				
	Interpretation	2	1	1	4	3	1	1	1	2	1
19	Evaluation of results and implications	1	1	1	1	1	1	1	1	1	1
20	Limitations of study	1			1	1				1	
21	Inferences				1	1					
22	Lesson learn				1						
	Conclusions and future work	2	2	2	3	2	2	2	1	2	1
23	Conclusions	1	1	1	1	1	1	1	1	1	1
24	Relation to existing evidence										
25	Impact										
26	Limitations				1						
27	Future work	1	1	1	1	1	1	1		1	
	Total	13	12	11	23	16	12	13	11	14	9

Tabla 6.4: Criterios para determinar la calidad de los experimentos

Continente	Citas	#Artículos
Europa	[9, 55, 72, 74, 90, 113, 116, 117, 124, 129, 135, 167, 185]	13
América del Norte	[57, 82, 94, 112, 169]	5
Asia	[35, 58, 132, 175]	4
No especificado	[20, 102, 103, 123, 151, 156, 184]	7

Tabla 6.5: Clasificación de los artículos basado en el continente

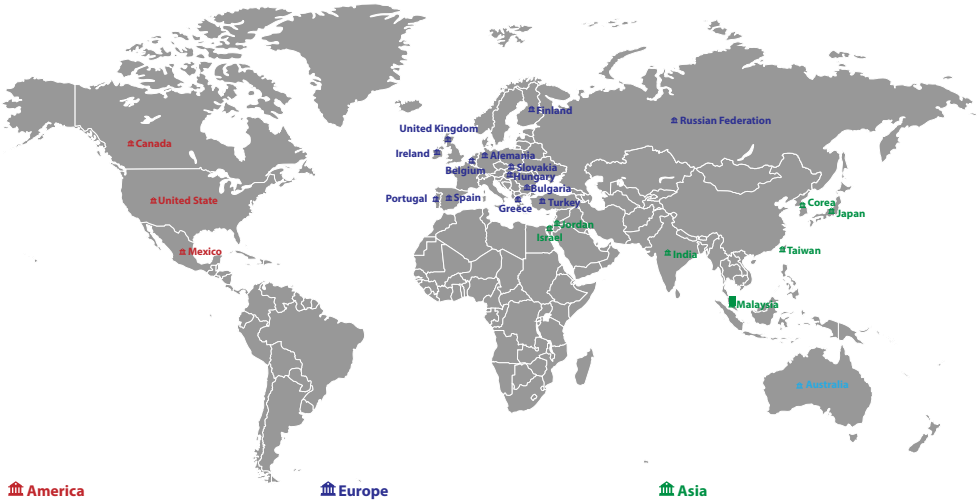

Figura 6.6: Clasificación de los artículos basado en institución-país-continente

responsable de la educación y la formación profesional en España[†9], con el número de trabajos publicados. El mayor número de estudios empíricos se ha realizado en la Educación Secundaria, con nueve estudios; a la educación secundaria le sigue de cerca la Educación Superior con ocho estudios. Esto

[†9]http://www.educacionyfp.gob.es/educacion-mecd/areas-educacion/estudiantes/portada.html

Nivel de educación	Citas	#Artículos
Educación Secundaria [12 a 18 años]	[57, 82, 90, 102, 112, 116, 117, 169, 184]	9
Educación Superior [mayor de 18 años]	[9, 55, 72, 74, 124, 132, 175, 185]	8
Educación Primaria [6 a 12 años]	[58, 113, 135]	3
Educación infantil [3 a 6 años]	[129]	1
No especificado	[20, 35, 94, 103, 123, 151, 156, 167]	8

Tabla 6.6: Clasificación de los artículos basados en el nivel de educación

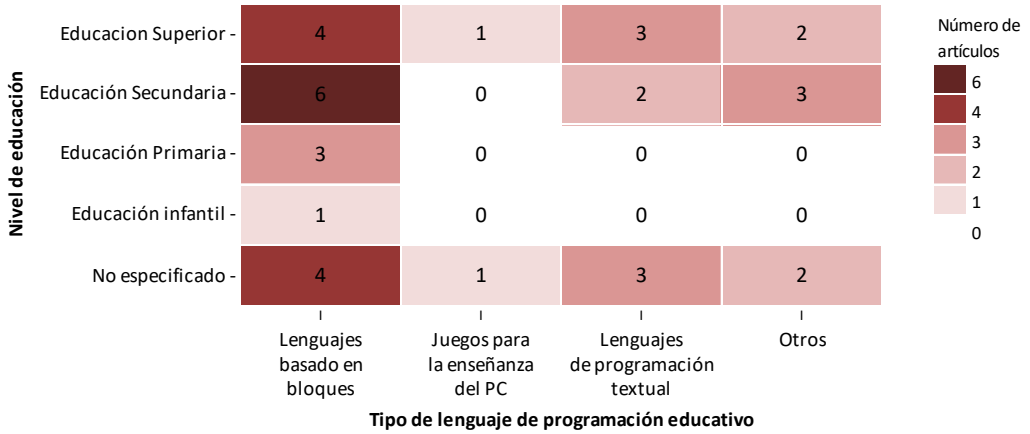

Figura 6.7: Visualización del mapa sistemático

podría deberse a que los estudiantes de mas de 18 años pueden tener cierta experiencia previa en conceptos de programación, lo que facilitaría la realización del experimento o del estudio de casos en niveles superiores.

Sin embargo, son pocos los estudios realizados en la Educación Primaria, con tres estudios y un estudio realizado en la educación infantil. Asimismo, hay 8 trabajos en los que no se especificó la etapa educativa en la que se realizó el estudio.

En la Figura §6.7 describen las lagunas de la investigación identificando los niveles educativos en los que se realizan los estudios empíricos y el tipo de lenguaje de programación educativo utilizado.

Actividad	Citas	#Artículos
Instrumento de recogida de datos		**12**
Cuestionario	[55, 113, 116, 184, 185]	5
Entrevistas	[72, 124, 151]	3
Encuestas	[132, 135, 156, 175]	4
Evaluación		**8**
Examen de programación	[58, 94, 102, 103, 129, 167, 169]	7
Auto-evaluación	[57]	1
Desarrollo de ejercicios	[20, 35, 90, 112, 117, 123]	**6**
Evaluación del proyecto final	[9, 74, 82, 113]	**4**

Tabla 6.7: Actividades para informar de la eficacia del LPE

6.3.3. Alternativas de evaluación de la eficacia del LPE

En esta sección se da respuesta a la pregunta de investigación

- *RQ3: ¿Como se reporta en la literatura la eficacia después de la aplicación de un Lenguaje de programación educativo?*

En el contexto de esta pregunta, se identifica la forma en que los autores informan sobre la eficacia tras la aplicación de los LPEs. Para responder a esto, solo se consideran las 29 artículos que presentan evidencia empírica. En [113] se reportan dos alternativas de evaluación de la efectividad, mientras que el resto de los autores solo reportaron una. La distribución de éstas actividades se presenta en la Tabla §6.7:

- **Instrumento de recogida de datos:** para esta categoría se consideraron las contribuciones que utilizaron I) Cuestionarios, II) Entrevistas y III) Encuestas , como técnicas de recolección de datos para conocer la opiniones de los estudiantes tras el uso del lenguaje de programación educativo.
- **Evaluación:** los trabajos de esta categoría incluyen actividades como: I) exámenes de programación, y II) autoevaluaciones, para determinar los logros en la comprensión y aprendizaje de los conceptos de programación en los participantes.
- **Desarrollo de ejercicios:** los trabajos de esta categoría analizaron las habilidades obtenidas en el desarrollo de ejercicios de programación para alcanzar los objetivos de aprendizaje propuestos en la aplicación de un lenguaje de programación educativo.

- **Evaluación del proyecto final:** los trabajos en esta categoría examinaron los proyectos diseñados y desarrollados por los participantes para evaluar el aprendizaje de los conceptos de programación.

De las actividades mencionadas en la Tabla §6.7, las más utilizada para determinar la efectividad de la aplicación de un LPE es el uso de instrumentos de recolección de datos. Ésta fue la técnica utilizada en 12 artículos de la literatura; se ha concluido que los cuestionarios, las entrevistas y las encuestas se consideran mecanismos básicos válidos, fiables y prácticos para obtener datos.

6.3.4. Objetivos pedagógicos con el uso de LPE

En esta sección se da respuesta a la pregunta de investigación:

- *RQ4: ¿Qué objetivos pedagógicos se logran con el uso de Lenguajes de programación educativos?*

Esta pregunta pretende determinar los objetivos pedagógicos en los estudios primarios. Para clasificar los objetivos pedagógicos en función de la adquisición de habilidades de programación, se analizaron tanto las 29 trabajos mencionados anteriormente como algunos otros que se mencionan a continuación para encontrar las clasificaciones existentes.

En primer lugar, la taxonomía de objetivos pedagógicos de Bloom y otros [23] fue creada en 1956 para promover altas formas de pensamiento en la educación. Sin embargo, en lo que respecta a la presente revisión bibliográfica, esta clasificación es muy genérica, ya que algunas habilidades que deben adquirirse al aprender lenguajes de programación no encajan exactamente en una categoría, sino que encajan en varias.

Es mejor ofrecer una clasificación mas acorde con las habilidades que puedan adquirir al aprender a programar. Para definir la mencionada clasificación, se recurrió a un análisis detallado y a la discusión de anteriores investigaciones sobre la enseñanza de la informática. Además de los 62 trabajos analizados es esta RSL, Midian Kurland y otros [104] realiza un estudio empírico en el que se identifica la falta de información tanto sobre las competencias alcanzadas como sobre la metodología que siguen los estudiantes para conseguirlas. La propuesta de [134] realiza una revisión de la literatura, que tiene como objetivo responder a las preguntas para descubrir el Conocimiento Pedagógico del Contenido de la Educación Informática. Y la contribución de [147] analiza muchos de los factores que influyen en el proceso de aprendizaje de la programación. Teniendo en cuenta todas las investigaciones, se identificaron, entre otras, las siguientes habilidades a adquirir:

1. El pensamiento algorítmico que implica secuencias, analizar y probar procesos en el tiempo y el espacio.
2. El juego creativo, la innovación y la exploración a través del entretenimiento, la comunicación y las aplicaciones sociales.
3. La aplicación de la creatividad y las habilidades adquiridas para resolver problemas.
4. El dominio de los conceptos y prácticas fundamentales.
5. El trabajo en equipo y la creación de asociaciones de apoyo.

Asimismo, los factores de motivación son necesarios a la hora de abordar temas asociados a los fundamentos de la programación dentro de los escenarios prácticos.

Por lo tanto, la clasificación final sería la siguiente, que son los cinco objetivos pedagógicos de alto nivel en el contexto de esta revisión sistemática de la literatura. La Tabla §6.8 presenta la distribución de los objetivos pedagógicos:

1. **Resolución de problemas:** consiste en la utilización de un proceso estructurado de resolución de problemas para ayudar a abordar nuevos problemas, ver las retos como problemas que pueden resolverse y ser capaces de descomponer los problemas mas grandes en elementos mas pequeños.

   Los trabajos clasificados en esta categoría plantean que la resolución de problemas computacionales ayuda a desarrollar habilidades para el aprendizaje de la programación como objetivo pedagógico. Esta categoría incluye elementos como I) el aprendizaje basado en problemas, II) comprensión del pensamiento computacional, III) aprendizaje de la programación informática; y, IV) la identificación de la lógica de un programa.

2. **Persistencia:** se fundamenta en considerar que los errores son una parte productiva y natural de la resolución de problemas, mantenerse firme y constante en la búsqueda de soluciones aunque se produzcan contratiempos, y realizar iteraciones para mejorar las soluciones parciales. Los trabajos en esta categoría proponen como objetivo pedagógico aumentar significativamente la constancia de los participantes en el aprendizaje y mejorar la comprensión de los conceptos de programación. Esta categoría incluye elementos como I) aumentar el rendimiento en la programación, II) mejora del aprendizaje de la programación, III) autoregulación; y, IV) aumento de la retención.

3. **Colaboración:** se basa en trabajar junto a otros para desarrollar mejores soluciones que incorporen el trabajo de todos; ayudar al equipo, inclu-

Objetivos pedagógicos	Citas	#Artículos
Resolución de problemas		**28**
Aprendizaje basado en problemas	[35, 113, 117, 124, 132, 151, 156, 169, 184, 185]	10
Comprensión del pensamiento computacional	[57, 82, 90, 112, 124, 129, 135, 169]	8
Aprendizaje de la programación informática	[57, 90, 94, 102, 103, 117, 123, 129, 135]	9
Identificación de la lógica de un programa	[151]	1
Persistencia		**11**
Aumentar el rendimiento en la programación	[9, 55, 58]	3
Mejoras en el aprendizaje de la programación	[9, 20, 55, 74, 116, 167]	6
Auto-regulación	[58]	1
Aumento de la retención	[9]	1
Colaboración	[57, 72, 82, 112, 185]	**5**
Creatividad		**4**
Creatividad e Innovación	[112, 185]	2
Razonamiento sistemático sostenido	[169, 185]	2
Comunicación	[55, 112]	**2**

Tabla 6.8: Objetivos pedagógicos reportados en la literatura

so mediando en los desacuerdos para conseguir una solución común; y contribuir activamente al éxito de los proyectos de grupo. Los trabajos de esta categoría establecen como objetivo pedagógico el trabajo en colaboración para ayudar a los participantes a desarrollar soluciones comunes aplicando conceptos de programación.

4. **Creatividad:** consiste en realizar el trabajo teniendo en cuenta sus propios intereses o ideas, considera varios enfoques posibles y experimenta con nuevas ideas, y construye, amplía e incluso mejora las ideas o proyectos de sus colegas. Los trabajos en esta categoría establecen como objetivo pedagógico el pensamiento creativo de los estudiantes para entender los conceptos de programación con una visión imaginativa. Esta categoría incluye elementos como I) creatividad e innovación; y, II) razonamiento sistemático sostenido.

5. **Comunicación:** se basa en estructurar y documentar el trabajo realizado

de forma que los demás puedan entenderlo fácilmente, ser empático con la audiencia en cuanto a su nivel de conocimiento al presentar el trabajo, y aceptar y proporcionar comentarios constructivos para mejorar el trabajo. Los artículos de esta categoría establecen como objetivo pedagógico la comunicación como forma de desarrollar un sentimiento de confianza en los participantes en su proceso de aprendizaje de la programación.

La motivación es un tema mencionado en la literatura que afecta a la consecución de los cinco objetivos mencionados. En Horváth y Javorský [72], los resultados de las entrevistas con los estudiantes sobre los métodos de aprendizaje utilizados establecen que la motivación es un problema que está parcialmente relacionado con el método de aprendizaje. Asimismo, Nikou y Economides [113] mencionan que la informática integra todas las disciplinas de la ingeniería; por lo tanto, todos los estudiantes necesitan estar motivados para aprender los principios básicos de la informática.

Sin embargo, es bastante difícil activar y mantener la motivación de los estudiantes para aprender a programar. Además, en los resultados de sus estudios Nikou y Economides [113], Reardon y Tangney [124] y Rozali y Zaid [132] establecieron que, a través la implementación del PBL (Aprendizaje basado en problemas), los estudiantes desarrollan habilidades de resolución de problemas y están motivados para aprender a programar.

6.3.5. Síntesis de otros resultados

La Tabla §6.9 muestra un resumen de las fuentes primarias con evidencias empíricas (29 artículos) junto con los datos relevantes extraídos en el proceso de revisión. Las columnas representan los autores clasificados por el método empírico utilizado en la investigación (Véase la Sección §6.3.1).

En primer lugar, se identifican los lenguajes de programación educativos mencionados en la literatura. Algunos autores utilizaron mas de un lenguaje de programación educativo en sus investigaciones. Entre los lenguajes mas utilizados por los autores, Scratch se utiliza aproximadamente el 24 % de las veces y le sigue AppInventor que se utiliza aproximadamente el 11 % de las veces. Esto puede ocurrir porque Scratch es una herramienta de fácil acceso y muy útil para iniciarse en la programación. La clasificación de los lenguajes utilizados se presenta en la Tabla §6.10:

- **Lenguaje basado en bloques:** los trabajos clasificados en esta categoría utilizan lenguajes de programación que permiten programar visualmente una serie de comandos secuencialmente, que descomponen bloques o piezas de código que se acoplan entre si.

	EXPERIMENTO													ESTUDIO DE CASO												OTROS			
	[9]	[20]	[55]	[58]	[94]	[102]	[123]	[129]	[167]	[185]	[35]	[82]	[90]	[103]	[112]	[113]	[116]	[117]	[124]	[135]	[151]	[156]	[169]	[175]	[184]	[57]	[72]	[74]	[132]
LENGUAJES BASADOS EN BLOQUES																													
Agent Cubes																							✓						
Alice																								✓					
App Inventor							✓									✓	✓	✓											
Erlang																												✓	
Piktomir								✓																					
Scratch				✓		✓				✓		✓		✓		✓			✓	✓	✓								
Snap																										✓			
Visual problem solving											✓																		
LENGUAJES DE PROGRAMACIÓN TEXTUAL																													
Action Script															✓														
Adobe Flash CS6																													✓
Java									✓										✓										
Language C													✓									✓							
Learning Objects																						✓							
Logo																											✓		
Python					✓																								
JUEGOS PARA EL DESARROLLO DEL PC																													
Gidget					✓																								
Second life		✓																											
OTRAS HERRAMIENTAS EDUCATIVAS																													
Flash design															✓														
Jeliot		✓							✓																				
LECGO													✓																
PILet	✓																												
RUR-PLE																									✓				
Unity																							✓						
OBJETIVOS DE INVESTIGACIÓN																													
Abordar los objetivos pedagógicos	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓			✓	✓	✓	✓
Evaluar un EPL						✓		✓												✓	✓				✓				
Incrementar la motivación					✓											✓				✓				✓			✓		✓
PC en el programa de estudios																				✓						✓			
OBJETIVOS PEDAGÓGICOS																													
Resolución de problemas					✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓		✓	✓			✓
Persistencia	✓	✓	✓	✓					✓								✓											✓	
Colaboración										✓		✓			✓											✓			
Creatividad										✓					✓								✓						
Comunicación			✓												✓														
Criterios de calidad (%)	48	44	41	85	59	44	48	41	52	33	80	60	67	60	73	53	53	33	60	60	33	40	33	60	40				
Nivel educativo	H		H	P		S		K		H		S	S		S	P	S	S	H	P			S	H	S	S	H	H	H

H:Educación superior S:Educación secundaria P:Educación primaria K:Jardín de infantes

Tabla 6.9: Síntesis de lenguajes de programación educativos y objetivos

Categorías	Citas	#Artículos
Lenguajes basados en bloques	[35, 57, 58, 74, 82, 90, 102, 103, 113, 116, 117, 123, 124, 129, 135, 151, 169, 175, 185]	19
Lenguajes de programación textual	[72, 90, 94, 112, 124, 132, 156, 167]	8
Juegos para la desarrollo del pensamiento computacional	[55, 94]	2
Otras herramientas educativas	[9, 20, 90, 112, 167, 175, 184]	7

Tabla 6.10: Clasificación de los artículos basados en el tipo de LPE

Objetivos	Citas	#Artículos
Abordar los objetivos pedagógicos	[9, 20, 35, 55, 57, 58, 72, 74, 82, 90, 94, 103, 112, 116, 117, 123, 124, 132, 135, 156, 167, 169, 185]	23
Evaluar un LPE o técnica usada	[102, 129, 135, 151, 184]	5
Incrementar la motivación en ciencias de la computación	[72, 113, 132, 135, 175]	5
Pensamiento computacional a lo largo del programa de estudio	[57, 135]	2

Tabla 6.11: Clasificación de los artículos basado en los objetivos del estudio

- **Lenguaje de programación de texto:** los trabajos clasificados en esta categoría utilizan lenguajes en los que es necesario escribir comandos con una sintaxis específica.
- **Juegos para desarrollar el pensamiento computacional:** los trabajos clasificados en esta categoría utilizan juegos de ordenador para el desarrollo del pensamiento computacional.
- **Otras herramientas educativas:** los trabajos clasificados en esta categoría utilizan herramientas diferentes a las anteriores para aprender a programar.

De entre las categorías mencionadas, se observa que la mayor cantidad de los estudios utilizan lenguajes basados en bloques para el aprendizaje de la programación (ocho contribuciones), seguidos de cerca por los lenguajes de programación textual (siete contribuciones). En consecuencia, no hay evidencia suficiente para establecer que tipo de LPE domina los procesos de aprendizaje. A partir de los datos obtenidos, parece que esto podría deberse al contexto y a los objetivos del estudio, ya que algunos lenguajes textuales sirven para un propósito y los lenguajes visuales para otro, aunque se suele argumentar que los lenguajes visuales sirven para introducir el aprendizaje de la programación textual a través del desarrollo de habilidades, como la creatividad, la capacidad de resolución de problemas y el trabajo en equipo.

Además, en la Tabla §6.11 se muestran los objetivos de investigación abordados por los autores (algunos de ellos tienen mas de uno). Cada objetivo se explica como sigue:

- **Abordar objetivos pedagógicos:** los trabajos categorizados como este objetivo utilizan algún lenguaje de programación educativo como alternativa a los lenguajes de programación convencionales.

- **Evaluar un LPE o una técnica utilizada:** los trabajos categorizados como este objetivo evalúan el uso de los LPE para su inclusión en el programa de estudio.
- **Aumentar la motivación en ciencias de la computación:** los trabajos clasificados como este objetivo pretenden promover las actitudes positivas de los estudiantes hacia la informática.
- **Pensamiento computacional en el programa de estudios:** los trabajos de esta categoría buscan identificar los beneficios de la implementación transversal de un LPE en el programa de estudio.

A partir de los objetivos mencionados, se observa que el aspecto mas importante considerado por los autores es atender a los objetivos pedagógicos. Esto significa que en 23 de los 29 artículos presentados en esta investigación (aproximadamente el 79 %), los autores buscan cumplir el objetivo pedagógico mediante el uso de un LPE en el proceso de aprendizaje de la programación. Esto puede deberse a que la mayoría de las personas que trabajan en esta área tienen como vocación la docencia y, por lo tanto, buscan que los alumnos mejoren sus calificaciones y alcancen las habilidades y conocimientos deseados de acuerdo al nivel.

En cuanto a los objetivos pedagógicos, el porcentaje de cumplimiento de criterios de calidad de los métodos empíricos y el nivel educativo observado en la Tabla §6.9, se analizan en la Sección §6.3.1.

6.4. Resumen

En este capítulo se presentó una revisión bibliográfica de las publicaciones científicas sobre el uso de los LPE para el aprendizaje de la programación. Se identificaron los objetivos pedagógicos que se intentan alcanzar mediante la aplicación de los LPE, el método para evaluar la eficacia de las herramientas, los principales objetivos de investigación que se persiguen en cada investigación en los últimos años y el tiempo de las aportaciones.

A través de la RSL, se verifica que la mayor cantidad de los estudios se han realizado en el nivel secundario aplicando un lenguaje de programación educativo considerando que ha tenido un efecto favorable en el desarrollo de las habilidades de aprendizaje de programación. Además, los resultados revelaron que los lenguajes de programación basados en comandos y bloques favorece el aprendizaje y permite el desarrollo de otro tipo de habilidades y destrezas además de la programación.

En concreto, se identificaron algunas oportunidades de investigación, como

I) utilizar diferentes alternativas a los lenguajes basados en bloques y textuales para comparar los resultados; II) contar con mas evidencia empírica y de mejor calidad; III) desarrollar estudios respecto al uso de lenguajes de programación textual en etapas educativas tempranas. El objetivo de este estudio fue orientar nuevas investigaciones utilizando estos hallazgos mas importantes que identifican futuras investigaciones analizando un lenguaje específico, como ALICE para establecer el impacto en el aprendizaje de la programación realizando experimentos controlados en un alto nivel educativo.

Capítulo 7

Uso de ALICE en el aprendizaje de la programación

La programación necesita ser puesta en contexto para que la gente entienda por qué lo están haciendo.

Caitlin Kelleher

Las tecnologías de la información y las comunicaciones (TIC) han tenido un impacto hacia todos los sectores que forman parte de la dinámica organizacional, incluyendo el entorno educativo, afectando los procesos de enseñanza y aprendizaje; así como la investigación y la gestión en general. En éste contexto globalizado, la adquisición de competencias como la programación se hace vital en las ciencias de la computación, donde cada día se presentan recursos de apoyo para garantizar este proceso.

En la Sección §7.1 se describen los aspectos generales de utilizar ALICE como herramienta de aprendizaje; más adelante en la Sección §7.2 se describe la metodología para llevar a cabo el experimento. Luego, en la Sección §7.3 se muestran los resultados empíricos del experimento realizado. Finalmente, resumimos este capítulo en la Sección §7.4.

7.1. Introducción

El aprendizaje inicial de la programación ha atraído el interés de los investigadores en virtud los problemas que presentan los estudiantes, ya que carecen de motivación e interés necesario para adquirir habilidades de programación, lo que nos lleva a plantearnos algunas dificultades, entre las cuales se encuentran I) la compleja sintaxis del primer lenguaje de programación con el que los estudiantes se contactan, II) la falta de habilidades de los estudiantes en resolución de problemas, razonamiento lógico y abstracto, que se necesitan en ésta área, III) la estrategia de memorización "pura"que los estudiantes utilizan para lograr el éxito en otras asignaturas, pero que no es suficiente en ésta área; y, IV) los inadecuados métodos de enseñanza.

Para enfrentar y atenuar estos problemas, existen otras alternativas para enseñar a programar. Algunos investigadores mencionan varias estrategias, de las cuales podemos mencionar I) entornos de programación visual basados en bloques, II) aprendizaje basado en juegos; y, III) *Massive Open Online Courses (MOOC)*.

En cuanto a la estrategia de utilizar una herramienta de programación visual para enseñar a programar, esta ha tenido impactos positivos en el rendimiento y actitud de los estudiantes hacia la programación. Se mencionan estudios donde se utilizaron lenguajes de programación visual, como ALICE en [7, 49], donde se estudia la actitud, rendimiento y satisfacción de los estudiantes universitarios en el proceso de aprendizaje de la programación; Scratch en [29], que analiza el rendimiento de los estudiantes universitarios en el primer nivel; y App Inventor en [18], donde se analiza y selecciona un lenguaje de programación basado en bloques para apoyar la enseñanza de algoritmos a los estudiantes que comienzan los cursos introductorios de una carrera de Ciencias de la Computación.

En la revisión de literatura realizada por Vinueza-Morales y otros [160] sobre la evidencia empírica referente al uso de los lenguajes de programación con fines educativos, los resultados del estudio demuestran de manera general la efectividad de usar lenguajes de programación visual para enseñar a programar. ALICE, Earlang y Scratch se destacan por su uso con estudiantes del nivel de educación superior. Este es uno de los motivos por el cual se seleccionó ALICE para el experimento realizado en el presente trabajo.

Por otra parte, en [161] se realiza una revisión de literatura de los trabajos que utilizaron ALICE para el aprendizaje de la programación. Se destaca que ALICE es útil para adquirir habilidades de resolución de problemas y desarrollo del pensamiento computacional. De los 24 artículos analizados, 14 de ellos se desarrollaron en un nivel universitario, lo que destaca el uso de ALICE en niveles superiores de educación. Esto ayudó a seleccionar el entorno ALICE

para su aplicación en el presente trabajo.

Como ha sido mencionado, una de las estrategias para enfrentar las dificultades en el proceso de enseñanza-aprendizaje de programación para los estudiantes es el entorno ALICE. La última versión de este lenguaje de programación es Alice 3[†1], el cual presenta características creativas con énfasis adicional en los conceptos orientados a objetos, y puede servir como una herramienta de aprendizaje e incluso puede ser extendida para ayudar a una transición al lenguaje de programación Java. Además, ofrece ALICE 3 Netbeans Plugin[†2], que es un entorno de desarrollo por módulos para crear aplicaciones en Java.

Utilizar ALICE que es considerada una herramienta orientada a objetos aumenta la confianza de los estudiantes y su capacidad en la comprensión de los conceptos de programación orientada a objetos [5]. El entorno de programación ALICE permite a los estudiantes crear vídeos animados, personajes o videojuegos mientras aprenden conceptos básicos de programación; los programas se crean arrastrando las instrucciones al área de edición, lo que también permite la construcción de interacciones entre objetos [42].

En un estudio realizado por [29], sobre la implicación se Scratch como herramienta para enseñar a programar a los estudiantes en nivel universitario, mencionó que uno de los problemas encontrados es sobre el nivel de aprobación de los estudiantes en la asignatura de programación, el cual no supera el 43 %; sin embargo, con la aplicación de la herramienta este porcentaje mejoró significativamente, es decir los estudiantes aprendieron a programar usando la herramienta educativa.

El objetivo del presente capítulo es determinar la efectividad del entorno ALICE para incrementar el desempeño en el aprendizaje de programación en estudiantes de la carrera de ingeniería en software de la Universidad Estatal de Milagro (UNEMI), en Ecuador, que cursan la asignatura Técnicas de Programación, donde deben aprender programación en Java. Para ello, se compara el éxito del aprendizaje mediante la técnica de enseñanza tradicional y mediante el uso de ALICE, con la idea de concluir cómo de beneficioso es su uso en nivel universitario.

7.2. Método

Siguiendo las directrices en Wohlin y otros [177], hay varias metodologías para llevar a cabo estudios empíricos como son encuestas, estudios de caso y experimentos, en el presente trabajo se realizó un experimento ya que se trabajó ba-

[†1] https://www.alice.org/get-alice/alice-3/
[†2] https://www.alice.org/get-alice/alice-3-with-netbeans/

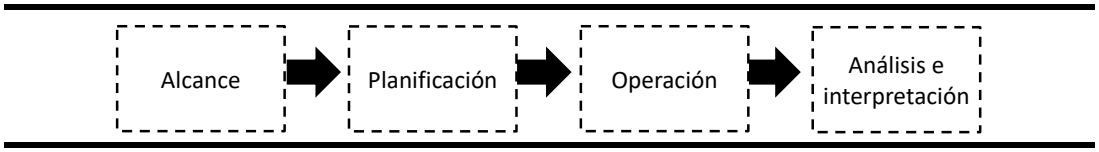

Figura 7.1: Fases del experimento

jo condiciones controladas con dos grupos de participante (experimental y de control) para establecer la efectividad del entorno ALICE en el aprendizaje de programación de los estudiantes.

El experimento está compuesto por cuatro fases, que se muestran en la Figura §7.1.

A continuación se detalla la realización de cada uno de estas fases:

7.2.1. Alcance

Debido a la dificultad en el aprendizaje de programación, derivando en el elevado índice de fracasos obtenidos por los estudiantes, nuestra investigación se enfocó en determinar la efectividad de la plataforma ALICE como herramienta para el aprendizaje de la programación. Dicho estudio se realizó a los estudiantes de ingeniería en software de la Universidad Estatal de Milagro, Ecuador.

Para lograr este objetivo, se realizó un trabajo de investigación mixta empleando técnicas cualitativas y cuantitativas. La estrategia cualitativa estuvo sustentada en la revisión de la literatura, así como observación. El enfoque cuantitativo se sustentó en el desarrollo de encuestas y el análisis estadístico de los datos generados a partir de la opinión de grupos de interés.

7.2.2. Planificación

En ésta fase del experimento tienen lugar todas las tareas relativas al diseño y planificación del estudio empírico. Para ello, los pasos seguidos son:

1. Selección del contexto: en este paso se estableció las dimensiones en las que se desarrolla el experimento, se identificó la situación actual previa al mismo y se concretó el estudio a abordar.

2. Formulación de hipótesis: se realizó la definición de las hipótesis que fueron comprobadas durante el desarrollo del experimento.

3. Selección de participantes: en este paso del proceso se seleccionó la población representativa para el estudio empírico.
4. Instrumentación: en este punto se determinó el material necesario para llevar a cabo el experimento, y finalmente se desarrollaron los instrumentos de medición y recogida de datos.
5. Evaluación de la validez: por último, se establecieron los procedimientos necesarios para determinar la validez de los resultados.

En las siguientes subsecciones de exponen en detalle cada uno de estos pasos.

7.2.2.1. Selección del contexto

El proceso de enseñanza-aprendizaje de la asignatura Técnicas de Programación se realiza mediante la forma tradicional, con clases teóricas y prácticas, donde el profesor explica de forma verbal los conceptos de programación y el estudiante debe repetir y memorizar la información. En el transcurso de la asignatura se realizan ejercicios prácticos utilizando el lenguaje de programación Java. El aprendizaje de los alumnos es medido por la cantidad de ejercicios resueltos por los estudiantes. Así mismo, al final de cada parcial, los estudiantes realizan un examen escrito de opción múltiple.

En el presente trabajo de investigación se realizó una comparativa del método tradicional de enseñanza de programación en el contexto educativo universitario, frente a un método de enseñanza que utiliza un lenguaje de programación educativo (en nuestro caso, ALICE), donde se pudo determinar el impacto en el conocimiento adquirido por los estudiantes que participaron en el experimento.

Se aplicó ALICE por ser una herramienta muy utilizada en el aprendizaje de programación, por ser muy útil para incrementar aptitudes en cuanto a resolver problemas y al desarrollo del pensamiento algorítmico. ALICE es una herramienta completamente accesible, mantiene varias versiones, la versión ALICE 3 Netbeans Plugin[†3] es un entorno de desarrollo modular diseñado principalmente para crear aplicaciones en Java, permitiendo cargar mundos desarrollados y almacenados en ALICE 3 y continuar editándolos directamente en el código Java. ALICE facilita la transición en codificación basada en texto hacia un lenguaje orientado a objetos como Java.

En [41] presenta un estudio donde compara la eficacia del uso de software ALICE con los lenguajes de programación convencionales en el ámbito del aprendizaje inicial de programación. Sin embargo, en el presente estudio se

[†3] https://www.alice.org/get-alice/alice-3-with-netbeans/

analizó el uso de ALICE para la comprensión de los conceptos orientados a objetos en la asignatura Técnicas de Programación donde los estudiantes debían aprender a programar en Java.

7.2.2.2. Formulación de hipótesis

Para el presente estudio, se plantea la siguiente hipótesis:

- **H1:** El fracaso en el aprendizaje de la programación en nivel universitario se puede reducir al apoyarse en un entorno de programación visual, como ALICE.

Adicionalmente se consideró para la presente contribución variables como el género, turno de estudio y la asistencia a clases donde se establecieron las incidencias con el aprendizaje de la programación.

7.2.2.3. Selección de participantes

El estudio se realizó en un ambiente universitario por el nivel de conocimiento previo que tienen los estudiantes lo que permitió obtener mejor provecho de la experiencia con ALICE, teniendo en cuenta que, para el nivel universitario es necesario que el estudiante tenga mayores destrezas en programación.

Concretamente, los participantes del proyecto eran estudiantes que cursaban la asignatura Técnicas de programación, del segundo nivel de la carrera de Ingeniería en software de la Universidad Estatal de Milagro, Milagro-Ecuador. Este curso es obligatorio para los estudiantes matriculados y es un requisito previo para sus otros cursos de programación avanzada. Los estudiantes necesitan que se cubra más a fondo los contenidos de la materia para prepararlos a sus cursos de programación de nivel superior.

En dicha asignatura los estudiantes aprenden a programar en Java, lo cual se considera difícil para los estudiantes por ser el primer curso de programación con conceptos orientados a objetos en su carrera. Por este motivo se seleccionó a los estudiantes del segundo nivel para introducir a los participantes en el aprendizaje de programación orientada a objetos mediante ALICE.

En nuestro experimento se trabajó con dos grupos, uno de control que vio la materia de programación de manera tradicional con el lenguaje de programación Java, y un grupo experimental que trabajó tanto con Java como con ALICE. Para conformar el grupo experimental se seleccionaron estudiantes de los dos turnos de estudio, matutino y nocturno, para que participen en el curso de ALICE como plataforma de aprendizaje, realizando posteriormente la división en grupo de control y experimental de manera aleatoria.

Grupo	Turno		Total
	Matutino	Nocturno	
Control	7	15	22
Experimental	4	16	20
Total	11	31	42

Tabla 7.1: Ficha muestral grupo/turno

Grupo	Género		Total
	Femenino	Masculino	
Control	9	13	22
Experimental	7	13	20
Total	16	26	42

Tabla 7.2: Ficha muestral grupo/género

En la Tabla §7.1, se presenta las características de la muestra empleada en la investigación. En este caso la cantidad de estudiantes correspondientes al turno de la mañana y la noche. El grupo de control, corresponde a los estudiantes que no cursaron programación con ALICE, mientras que el grupo experimental sí.

En la Tabla §7.2, se presenta la muestra empleada en la investigación, considerando el género que corresponde a cada estudiante, en la Sección de resultados §7.3 se realiza la incidencia del género en el aprendizaje de la programación.

7.2.2.4. Instrumentación

Una vez establecido el propósito de la presente contribución, se identifica el tipo de evaluación empírica que cubre las necesidades, de entre: (I) encuestas, método que se utiliza para obtener datos que están en la memoria de los entrevistados; (II) estudios de caso, utilizados para identificar y documentar causas clave que incidan en los resultados de una actividad; y, (III) experimentos formales, que se emplean para, de forma controlada y rigurosa, indagar aquellos causantes que afectan el tema a investigar.

En nuestro caso, la realización de encuestas fue seleccionada como tipo de evaluación empírica, por considerarse un método cuya intención es documentar relaciones y resultados, así como recabar datos para ser analizados de forma

cuantitativa.

A su vez, y dentro de los tipos de encuestas que se pueden realizar (estructuradas, no estructuradas y semi-estructuradas), utilizamos encuestas estructuradas, que constan de preguntas fijadas de antemano, iguales para todos los participantes involucrados y que no sufren alteraciones durante la entrevista. Su uso nos permite que las conclusiones puedan generalizarse, teniendo en cuenta, tanto la población objeto de estudio como el porcentaje de respuestas esperadas.

Concretamente, para la realización de esta contribución se diseñaron dos cuestionarios:

- **Cuestionario A**, cuya cumplimentación se realizó por parte de todos los estudiantes (ambos grupos, control y experimental) y se realizó antes de comenzar la ejecución del experimento. Su objetivo es consultar la opinión de todos los estudiantes sobre aspectos generales de programación. Dicho cuestionario se presenta en la Tabla §7.3, y consta de cuatro partes:

  I) parte a: datos generales, como edad, género, calificación previa en programación, o promedio del primer semestre, turno de estudio, trabaja; II) parte b: nivel de conocimiento de ALICE, Scratch, Java, C, C++; III) parte c: nivel de conocimiento en conceptos de programación; y, IV) parte d: otros aspectos, como cumplimiento de tareas, asistencia a clases, capacidad para superar la asignatura y perspectiva de aprobación de la asignatura Técnicas de Programación, con el propósito de apoyar o refutar la hipótesis establecida en el planteamiento del problema, y que pudiera afectar los resultados de la investigación.

- **Cuestionario B**, cuya aplicación tiene lugar después de la realización del experimento utilizando ALICE, y sólo al grupo experimental. Dicho cuestionario, presentado en la Tabla §7.4, tiene como objetivo recabar datos acerca de la percepción y conocimientos adquiridos en la experiencia con ALICE, y consta de tres partes:

  I) parte a: percepción sobre la experiencia con ALICE; II) parte b: percepción del nivel de conocimiento en conceptos de programación; y, III) parte c: percepción del grado en el cual ALICE le ayudó a entender los conceptos de programación.

7.2.2.5. Evaluación de la validez

Para validar los resultados presentados en esta contribución, se consideró lo siguiente:

Parte a
Datos generales
Edad | Género | Calificación Fundamentos de Programación | Promedio semestre 1 Turno de estudio | Trabaja
Parte b
Nivel más alto que ha estudiado los siguientes elementos de Programación
(Opciones: Universidad | Colegio | Auto-aprendizaje | No he estudiado) Alice | Scratch | Java | C | C++ | Otro
Parte c
Nivel de conocimiento en conceptos de programación
(Escala 1-2-3-4-5 | 1, Nada, 2, Poco, 3, Lo suficiente, 4, Bastante, 5, Mucho) Variables | Objetos | Clases | Procedimientos | Funciones | Métodos en general Parámetros | Condicionales | Bucles | Arreglos | Programación en general
Parte d
Otros aspectos
(Opciones 1-2-3-4-5 | 1, Totalmente en desacuerdo, 2, Desacuerdo, 3, Ni de acuerdo ni en desacuerdo, 4, De acuerdo, 5, Totalmente de acuerdo) d.a) Cumplimiento con tareas encomendadas por el docente d.b) Asistencia a clases d.c) Capacidad intelectual para aprobar la asignatura Técnicas de Programación d.d) Perspectiva de aprobación de la asignatura de Técnicas de Programación

Tabla 7.3: Cuestionario A

- **Validación externa.** Como se mencionó anteriormente, para garantizar la representatividad de la población durante el estudio, aplicamos una selección de estudiantes al azar. Los estudiantes tenían una matrícula regular, es decir cumplían con los requisitos de estudiantes regulares.

  Todos tenían conocimientos básicos de programación. Para valorar la experiencia de los participantes en el experimento se aplicaron varios cuestionarios para conocer la experiencia del aprendizaje mediante ALICE.

- **Validación interna.** En este estudio experimental se intentó ser lo más exhaustivo metodológicamente posible. Con escenarios de aprendizaje similares para ambos grupos de tal manera que no se afecte el proceso de aprendizaje. Grupos formados por estudiantes con características cognitivas similares ya que se encuentran en el mismo nivel de estudio y con una edad promedio global de 20 años.

  Los datos recogidos mediante las encuestas son confiables, ya que se ob-

Parte a
Percepción sobre la experiencia con ALICE
(Opciones 1-2-3-4-5 | 1, Totalmente en desacuerdo, 2, Desacuerdo, 3, Ni de acuerdo ni en desacuerdo, 4, De acuerdo, 5, Totalmente de acuerdo) - ALICE me ayudó a entender los conceptos de programación de computadoras. - Encontré a ALICE interesante y fácil de usar. - Tengo más posibilidades de ser promovido si se utiliza ALICE como parte de la asignatura Técnicas de Programación. - Considero que ALICE contribuirá a mejorar mis calificaciones en la asignatura Técnicas de Programación. - Considero que ALICE podría ser utilizado como parte de las estrategias utilizadas por el docente de la asignatura Técnicas de Programación. - Me gustaría asistir a cursos donde se profundicen los conceptos y usos de la herramienta ALICE. - La inclusión de ALICE en el contenido de Técnicas de Programación cambió positivamente mi perspectiva de la materia. - Creo que utilizar ALICE frente a otras estrategias de enseñanza puede mejorar las calificaciones de los estudiantes de la asignatura Técnicas de Programación.
Parte b
Percepción del nivel de conocimiento en conceptos de programación
(Escala 1-2-3-4-5 | 1, Nada, 2, Poco, 3, Lo suficiente, 4, Bastante, 5, Mucho) Variables | Objetos | Clases | Procedimientos | Funciones | Métodos en general Parámetros | Condicionales | Bucles | Arreglos | Programación en general
Parte c
Percepción del grado en el cual ALICE le ayudó a entender los siguientes conceptos de programación (Escala 1-2-3-4-5 | 1, Nada, 2, Poco, 3, Lo suficiente, 4, Bastante, 5, Mucho) Variables | Objetos | Clases | Procedimientos | Funciones | Métodos en general Parámetros | Condicionales | Bucles | Arreglos

Tabla 7.4: Cuestionario B

tuvieron cuando los estudiantes asistieron a clases de manera presencial, con eso nos aseguramos que han formado parte del estudio, además que son anónimas para evitar datos falseados.

7.2.3. Operación

Como ya ha sido mencionado, la realización del estudio empírico tuvo lugar en el segundo parcial del curso. De este modo, durante el primer parcial los estudiantes cursaron la materia de la manera tradicional, y fue en el segundo parcial cuando se dividió el curso en los grupos experimental y de control, se realizó la experiencia con ALICE y la recopilación de datos mediante las encuestas.

En detalle, en la Figura §7.2 se observa el proceso desarrollado y consta de las siguientes fases:

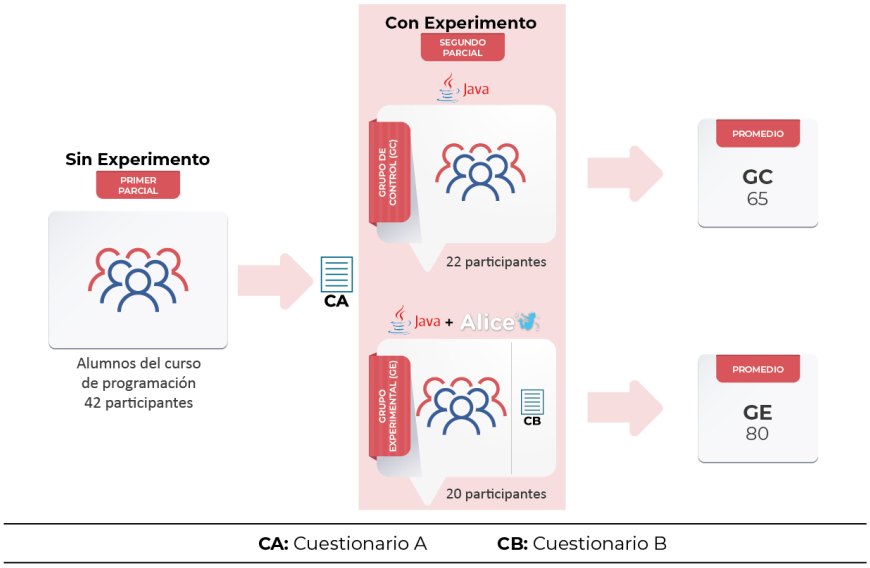

Figura 7.2: Proceso general

1. Durante el segundo semestre de la carrera de ingeniería en software, los estudiantes de ambos turnos (matutino y nocturno), en el contenido de la asignatura Técnicas de programación, realizan el aprendizaje y evaluación de la asignatura de la manera tradicional. Para ello, reciben clases teóricas y prácticas de Java, para posteriormente rendir el examen escrito oficial de la asignatura al finalizar el primer parcial. Dicho examen, junto con otras actividades (como por ejemplo talleres individuales y grupales, tareas, entre otras), completan los 50 puntos correspondientes a la calificación del primer parcial (P1).

2. Tras la evaluación del primer parcial, y antes de comenzar el experimento con ALICE, se aplicó el Cuestionario A (ver Tabla §7.3) a todos los estudiantes del segundo semestre de la asignatura Técnicas de Progra-

mación. De esta forma se establece cuánto conocen sobre algunos aspectos generales de programación.

3. A la hora de comenzar el estudio, se establecieron dos grupos de trabajo, el grupo de control y el grupo experimental. Considerando que la población estaba conformado por 42 estudiantes, se crearon dos grupos, el grupo de control con 22 integrantes y para el grupo experimental 20 integrantes, seleccionados de forma aleatoria, tal y como se describe en la Sección §7.2.2.3.

4. Tras esto, en paralelo, los estudiantes del grupo de control continúan el aprendizaje y evaluación de la manera tradicional, mientras que los integrantes del grupo experimental reciben formación mediante la herramienta ALICE.

5. Una vez desarrollada la experiencia con ALICE, se aplica el Cuestionario B (ver Tabla §7.4) solamente al grupo experimental, para conocer opiniones sobre este lenguaje de programación educativo.

6. Finalmente, los estudiantes de ambos grupos completan su evaluación formal escrita de la asignatura Técnicas de Programación, del segundo parcial (P2), para acumular 50 puntos. Se presenta en la Figura §7.2 los promedios (sobre 100 puntos) obtenidos de ambos grupos, el grupo de control alcanzó un promedio en notas de 65 puntos, en tanto que el grupo experimental obtuvo un promedio de 80 puntos.

7.2.4. Análisis e interpretación

En la presente sección se expone el análisis de los datos obtenidos en la presente contribución, los mismos que son detallados en la siguiente sección (Ver Sección §7.3).

7.3. Resultados

En la presente sección se muestran los resultados del análisis realizado a los datos que corresponden a las calificaciones de los estudiantes obtenidos durante la realización del experimento y a los datos recabados mediante la aplicación de los cuestionarios. Para acceder a los datos, revisar el repositorio [†4].

[†4] https://github.com/MariuxiVinueza/tesis_evidencia_empirica_epl.git

7.3.1. Resultados del análisis de las calificaciones

Para reportar los resultados del experimento en cuanto a las calificaciones, se utilizaron técnicas de minería de datos, cuyo proceso consiste en extraer y procesar información para obtener tendencias y patrones de los datos, también para describir la relación entre los atributos, que en este caso serían las notas del grupo experimental y del grupo de control.

Las técnicas de minería de datos son procesos que nos ayudan a entender nuevas matrices de datos y encontrar finalmente su información implícita, la cual no es posible obtener mediante el uso de métodos estadísticos convencionales. Estas técnicas permiten buscar patrones que se encuentran incluidos en una gran cantidad de información, las técnicas más usadas en el campo de la ingeniería son: los algoritmos PCA Biplot y las reglas de asociación [56].

Dichas técnicas de minería de datos fueron utilizadas para el presente estudio, para analizar las calificaciones de los alumnos y establecer que el uso de un lenguaje de programación educativo reduce el fracaso en el aprendizaje de la programación, como se indica en la hipótesis del presente estudio (ver Sección §7.2.2.2).

7.3.1.1. Algoritmo PCA Biplot para diferentes lenguajes de programación educativa

Las gráficas en las Figuras §7.3 y §7.4 fueron realizadas utilizando el algoritmo de PCA Biplot. En ellas, los ejes X e Y representan una confiabilidad del 100 % en la distribución de los grupos que corresponden a las calificaciones del primer parcial, segundo parcial y nota final. Estas gráficas permiten evidenciar que el uso de clusters o agrupamientos son necesarios para indicar que sí hubo diferencias cuando se utilizaron los dos tipos de lenguajes en la enseñanza de programación a los estudiantes.

La gráfica en la Figura §7.3 presenta la mayor distribución de las calificaciones en el primero y cuarto cuadrante, indicando una misma relación en el aprendizaje. Es decir, se obtuvieron similares calificaciones en el primer y segundo parcial utilizando el lenguaje Java.

Por otra parte, ángulo que presenta en el primer cuadrante la gráfica en la Figura §7.4 nos indica una alta relación entre las notas del segundo parcial y las notas finales, presentando una gran diferencia con el uso de lenguaje ALICE en relación al lenguaje Java que se utilizó en el primer parcial, y cuyo ángulo en la figura presenta una relación lenta en el aprendizaje. En base a los resultados, el uso del lenguaje ALICE proporciona una mejor técnica para explicar los conceptos de programación a los estudiantes.

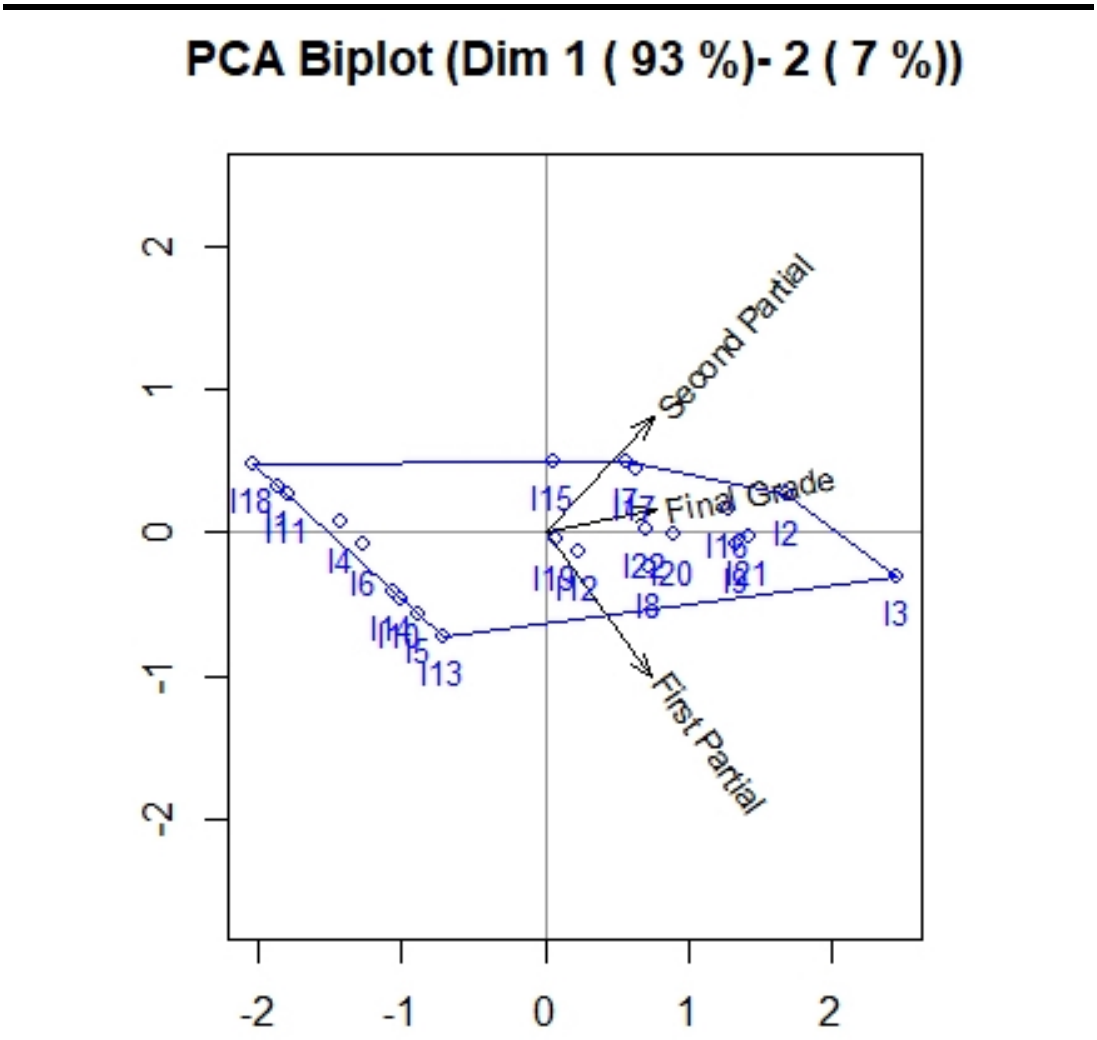

Figura 7.3: Algoritmo PCA Biplot utilizando Java en todo el semestre

7.3.1.2. Algoritmo de reglas de asociación para diferentes lenguajes de programación educativos

El programa de asociaciones está basado en el ingreso de las calificaciones del primero y segundo parcial, el turno de estudio y el género del estudiante, este algoritmo presenta la relación que existe entre los datos ingresados generando una figura donde se ve la relación entre ellas.

Las Figuras §7.5 y §7.6 presentan el uso de reglas de asociación para conjuntos de datos que representan los lenguajes de programación Java y ALICE. El programa de reglas de asociación es una solución para extraer reglas alternativas, ya que proporciona una imagen completa de las asociaciones en un gran conjunto de datos.

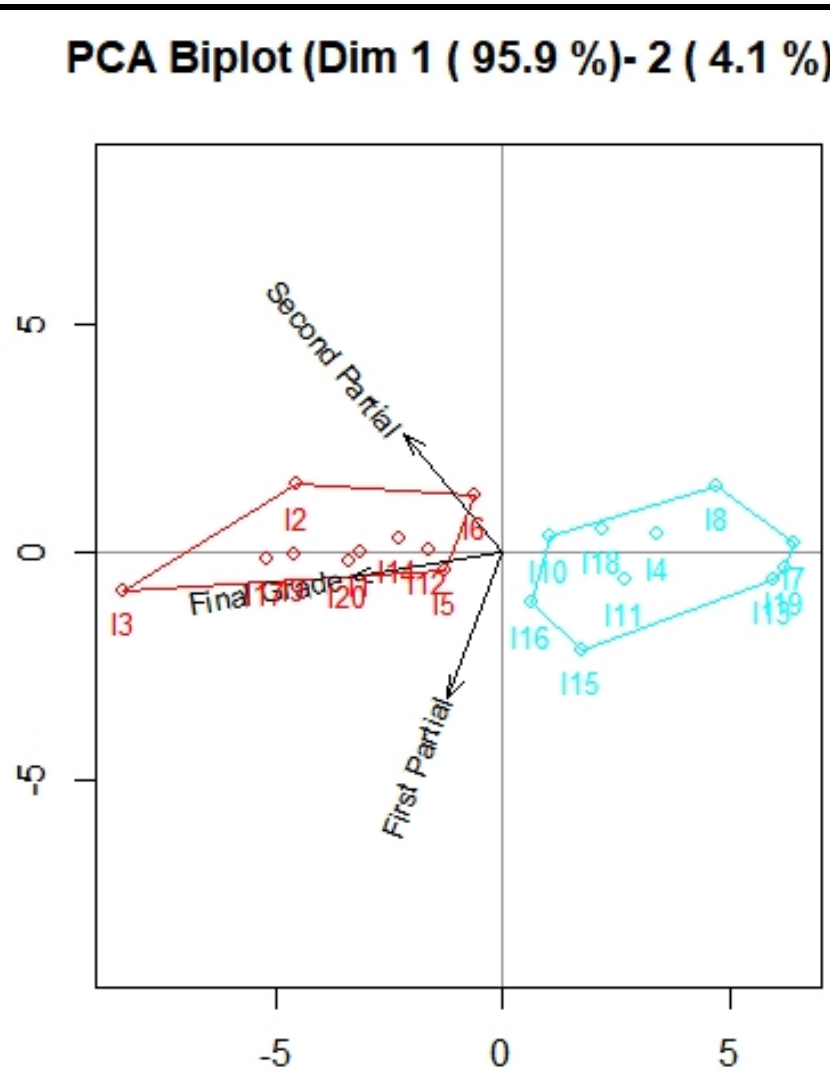

Figura 7.4: Algoritmo PCA Biplot para utilizando Java en primer parcial y ALICE en segundo parcial.

Ambas figura §7.5 y §7.6, muestran que las calificaciones finales de los estudiantes en la asignatura de Técnicas de programación, utilizando los lenguajes Java y Alice, están influenciadas por el género y también por el turno de estudio.

7.3.2. Resultados del análisis de los cuestionarios

En esta sección se presentan los resultados obtenidos a través de los cuestionarios realizados a los estudiantes. Como ya se explicó, antes de la realización

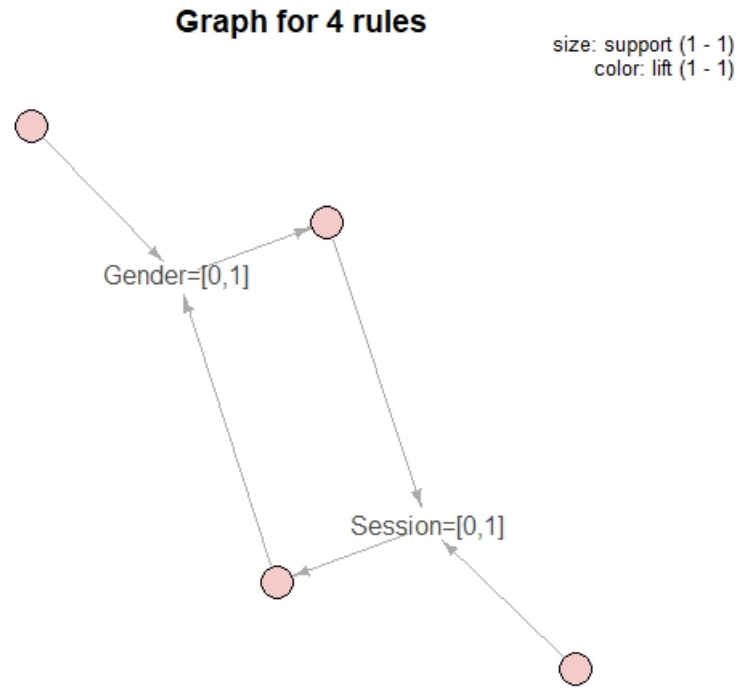

Figura 7.5: Algoritmo de reglas de asociación para las calificaciones con Java en todo el semestre.

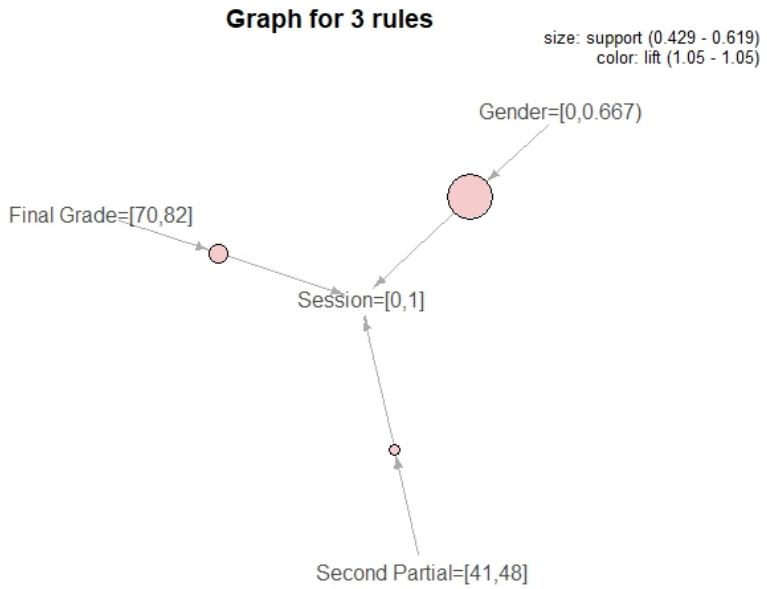

Figura 7.6: Algoritmo de reglas de asociación para las calificaciones con Java en el primer parcial y el lenguaje ALICE en el segundo parcial.

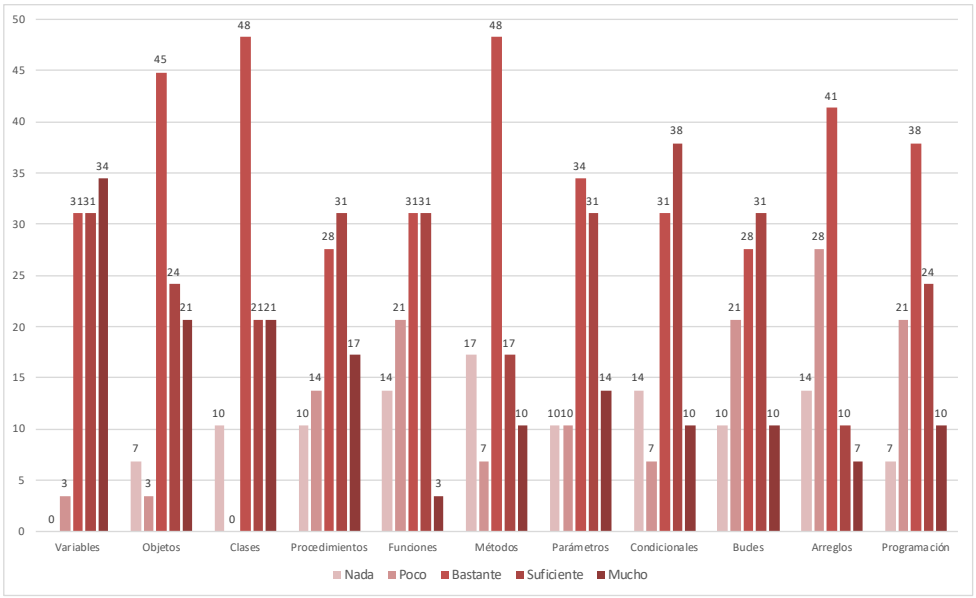

Figura 7.7: Nivel de conocimientos en conceptos de programación

del curso de ALICE, se aplicó el ***Cuestionario A*** a todos los estudiantes del segundo semestre de la carrera de Ingeniería en software. Los resultados sobre el nivel de conocimientos en conceptos de programación, **parte c**, se presentan en la Figura §7.7, donde se evidencia el alto porcentaje de conocimientos en general, en promedio más del 40 % de los estudiantes manifiestan tener *bastante/mucho* conocimiento de conceptos básicos de programación.

Aunque ningún estudiante manifestó conocer ALICE, los conocimientos en programación evidenciados por los resultados de la Figura §7.7 se respaldan con el hecho de haber estudiado programación básica antes, 34 % Scratch, 100 % Java, 21 % C y 76 % C++, manifestado así por los estudiantes en la **parte b** del ***Cuestionario A***.

Entre otros aspectos recogidos con el ***Cuestionario A*** y que se presentan en la Figura §7.8, de la **parte d** se destaca que casi un 80 % manifiesta que está *de acuerdo/totalmente de acuerdo* en que cumple con regularidad las tareas asignadas por el docente; mas del 80 % asiste frecuentemente a clases; mas del 70 % considera que tiene la capacidad intelectual suficiente para superar la asignatura Técnicas de Programación; y más del 70 % de los estudiantes cree que aprobará la asignatura Técnicas de Programación.

Con los datos disponibles, y con el propósito de descartar variables que pudieran influir significativamente en los resultados del estudio, se evaluaron aspectos relacionados con el género y el turno de clases. En cuanto al género, se

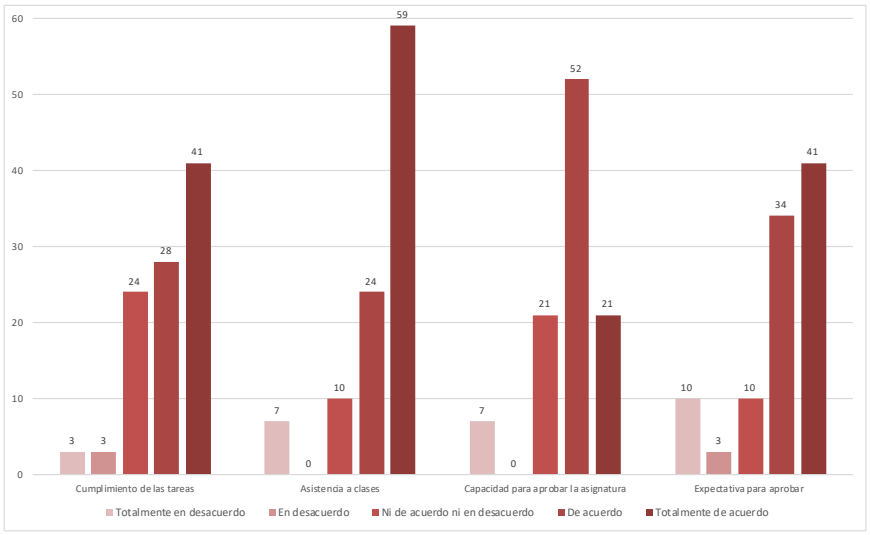

Figura 7.8: Otros aspectos sobre actitud/aptitud en programación

compararon los promedios de las notas de los grupos, femenino y masculino, donde se presenta diferencias significativas entre ellos (los valores de *Sig.*, no son menores a 0,05; establecido 5 % como nivel de significación para la prueba), por lo que se evidencia el género como posible influencia en los resultados de este estudio. En la Tabla §7.5, se reportan los resultados del análisis de la varianza (ANOVA) para comparar los promedios de notas (Prueba 1, Prueba 2 y Nota Final) de los grupos por género, información que fue proporcionada por el profesor responsable de la asignatura Técnicas de Programación.

En cuanto al turno en que el estudiante asiste a clases, se compararon los promedios de notas del turno matutino y nocturno, y tal como se presenta en la Tabla §7.6, se evidencia diferencias significativas entre los grupos (los valores de *Sig.* no son menores a 0,05; establecido como nivel de significación para la prueba).

Otro factor importante para valorar el desempeño de los estudiantes en el curso de Técnicas de Programación, es la asistencia regular a clases. Considerando los resultados mostrados en la Tabla §7.7, el coeficiente de correlación r=0,726, significativo al 5 %, evidencia que la asistencia a clases es un factor que incide positivamente en los resultados del desempeño del estudiante.

En la Tabla §7.8, se muestran los resultados de la comparación del desempeño promedio entre los grupos experimental y de control, donde se evidencian diferencias significativas entre los grupos; sin embargo, dado que también existe diferencia relevantes entre los grupos en la primera evaluación, ésta diferencia pudo haber sido por la condición particular de cada estudiante, por lo que se

		Suma de cuadrados	Grados de libertad	Media cuadrática	F	Sig.
Prueba 1	Género	273,000	1	273,000	3,595	,065
	error	3037,500	40	75,938		
	Total	3310,500	41			
Prueba 2	Género	659,297	1	659,297	2,277	,139
	error	11582,322	40	289,558		
	Total	12241,619	41			
Nota Final	Género	1780,797	1	1780,797	3,075	,087
	error	23167,322	40	579,183		
	Total	24848,119	41			

Tabla 7.5: ANOVA género

		Suma de cuadrados	Grados de libertad	Media cuadrática	F	Sig.
Prueba 1	Turno	5,204	1	5,204	,063	,803
	error	3305,296	40	82,632		
	Total	3310,500	41			
Prueba 2	Turno	41,613	1	41,613	,136	,714
	error	12200,006	40	305,000		
	Total	12241,619	41			
Nota Final	Turno	17,386	1	17,386	,028	,868
	error	24930,733	40	623,268		
	Total	24948,119	41			

Tabla 7.6: ANOVA turno de clases

			PA
Rho de Spearman	Nota Final	Coeficiente de correlación	,726
		Sig. (bilateral)	,000
		N	42

Tabla 7.7: Estadístico de correlación desempeño vs asistencia

		Suma de cuadrados	Grados de libertad	Media cuadrática	F	Sig.
Prueba 1	Grupos	321,109	1	321,109	4,297	,045
	error	2989,391	40	74,735		
	Total	3310,500	41			
Prueba 2	Grupos	1958,455	1	1958,455	7,618	,009
	error	10283,164	40	257,079		
	Total	12241,619	41			
Nota Final	Grupos	3865,601	1	3865,601	7,334	,010
	error	21082,518	40	527,063		
	Total	24948,119	41			

Tabla 7.8: ANOVA para desempeño según grupo

	Grupo	N	Media	Desviación Típica	Error tip. de la media
Dt	Experimental	20	8,000	11,36940	2,54227
	Control	22	-,1364	11,96541	2,55103

Tabla 7.9: Estadísticos según grupo

requiere una prueba de comparaciones de media que tome en cuenta la diferencia entre las calificaciones, una *prueba t* para los datos pareados.

Dado que se supone que los resultados del desempeño de los estudiantes en la Prueba 2, están determinados por los resultados de la Prueba 1, se realiza una prueba estadística para datos pareados, donde se busca evidenciar si existen diferencias significativas entre la mejora del desempeño promedio, calificación prueba 2 - calificación prueba 1 = Δt, cuando los datos están relacionados. En la Tabla §7.9, se presentan los estadísticos correspondientes a las diferencias entre las calificaciones, antes (P1) y después (P2) del taller. En la Tabla §7.10, se evidencian los resultados de la *prueba t* para datos pareados.

Dado los resultados de la *prueba t*, presentados en la Tabla §7.10, significativos al 5 %, se evidencia que existen diferencias significativas entre los grupos experimentales, con ALICE, y el grupo de control que aplica el enfoque tradicional.

Adicionalmente, se aplicó el Cuestionario B, a los participantes que cursa-

	t	gl	Sig.
Δt	2,254	40	0,0030

Tabla 7.10: Prueba T para igualdad de medias

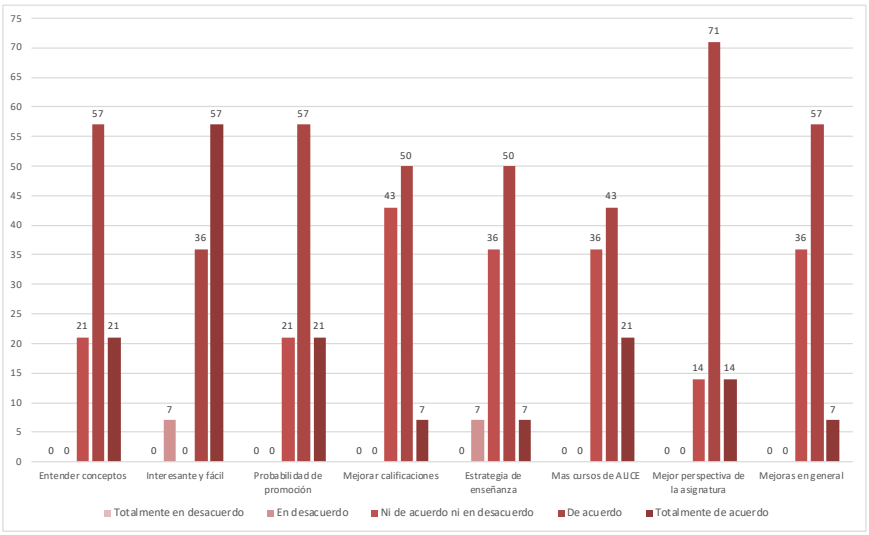

Figura 7.9: Percepción sobre la experiencia con ALICE

ron programación con ALICE; cuyo propósito era conocer la percepción que tenían sobre la experiencia y conocimientos adquiridos. En la Figura §7.9, se muestran los resultados de la **parte a** del Cuestionario B, donde se destaca que casi un 80 % de los estudiantes tienen la percepción de que ALICE ayudó en la comprensión de los conceptos de programación de computadoras, y, más del 90 % encontró ALICE interesante y fácil de usar.

Alrededor del 80 % está *de acuerdo/completamente de acuerdo* en que tiene más posibilidades de ser promovido si se utiliza ALICE como parte de la asignatura de Técnicas de Programación; mientras que un 57 % considera que ALICE contribuirá a mejorar sus calificaciones en la asignatura Técnicas de Programación, y que ALICE podría ser utilizado como parte de las estrategias empleadas por el docente de la asignatura Técnicas de Programación. A mas del 60 % de los estudiantes que cursaron ALICE, le gustaría asistir a cursos donde se profundicen conceptos y usos de la herramienta ALICE, y cree que utilizar ALICE frente a otras estrategias de enseñanza puede mejorar sus calificaciones de la asignatura Técnicas de Programación. Finalmente, se destaca en ésta **parte a** del cuestionario que casi el 90 % de los estudiantes es de la opinión de que la inclusión de ALICE en el contenido de Técnicas de Progra-

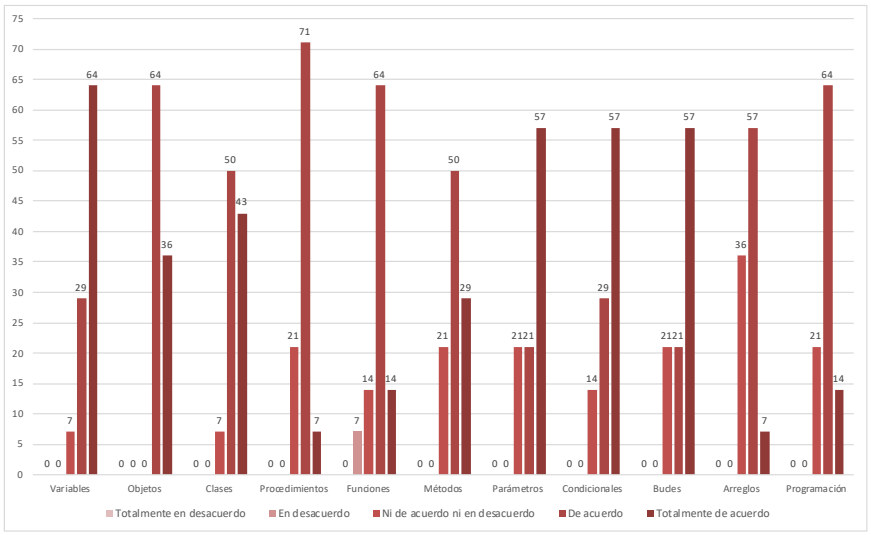

Figura 7.10: Percepción del nivel de conocimientos en conceptos de programación con ALICE

mación cambió positivamente su perspectiva de la materia.

Como se evidencia en la Figura §7.10, sobre la percepción del nivel de conocimientos que manifiestan tener los estudiantes en Programación, mas del 80 % en promedio considera que tienen conocimientos en programación; lo que representa una mejora significativa, si se compara con la percepción antes de cursar ALICE, representado en la Figura §7.7.

Adicionalmente, en la Figura §7.11, se evidencia que mas del 87 % en promedio, considera que ALICE le ayudó a entender los conceptos de programación.

7.4. Resumen

A partir de los resultados de la experiencia con estudiantes de Ingeniería en software de una universidad ecuatoriana, con respecto a la hipótesis **H1** se concluye que existe una reducción de fracasos en el aprendizaje de la programación, comparando los promedios de notas entre el primero y segundo parcial existiendo un incremento en las notas luego que los estudiantes aprendieron programación con ALICE en el segundo parcial.

No se evidenciaron diferencias significativas al comparar los promedios de notas entre los grupos en función del género (femenino, masculino) como se presenta en la Tabla §7.5. En cuanto a la comparación los promedios de notas

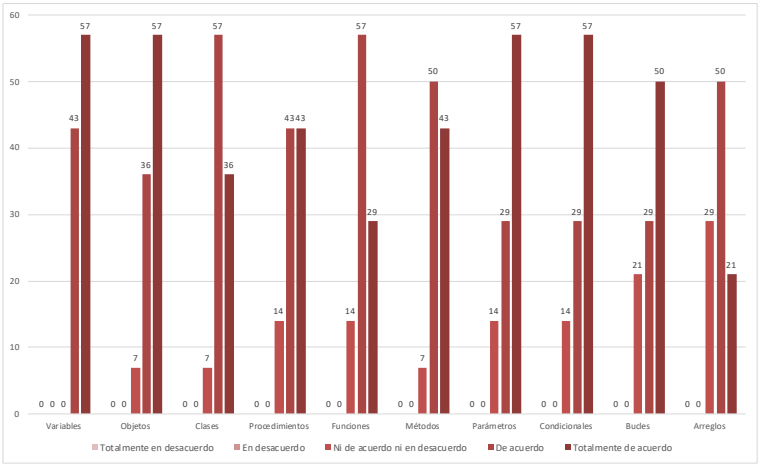

Figura 7.11: Percepción del grado en el cual ALICE ayudó a entender conceptos de programación

de cada turno (matutino y nocturno) como se muestra en la Tabla §7.6 concluyendo que igualmente no se encontraron diferencias significativas.

Por otro lado, se evidencia la influencia positiva que tiene la asistencia regular a clases sobre los resultados del desempeño académico, mostrado en Tabla §7.7. Así mismo, se evidencian mejores resultados en el desempeño académico en el grupo experimental que participó en el taller ALICE con respecto a los que aprendieron programación de la manera tradicional (hipótesis **H1**). La Tabla §7.8 presenta resultados de la comparación del desempeño promedio entre los grupos de control y experimental.

Parte IV

Conclusiones finales

Capítulo 8

Conclusiones y trabajo futuro

El papel del profesor es crear las condiciones para la invención, en lugar de proporcionar un conocimiento ya hecho.

Seymour Papert

En esta tesis se presenta un análisis de las evidencias empíricas de la aplicación de los diferentes lenguajes de programación educativos en el contexto educativo para el aprendizaje de la programación y su aplicación en el nivel universitario. Este capítulo presenta las conclusiones y el trabajo futuro basado en los resultados obtenidos.

8.1. Conclusiones

En esta tesis hemos demostrado que:

> Los lenguajes de programación educativos tienen una incidencia positiva en el proceso de aprendizaje de la programación, específicamente el entorno de programación ALICE, que fue aplicado en un contexto universitario logrando que los estudiantes aprendieran conceptos de programación orientada a objetos de forma efectiva.

En el presente trabajo, se revisaron y evaluaron artículos sobre la evidencia empírica del uso de los programas educativos para el aprendizaje de la programación, en estudiantes de todos los niveles educativos y que fueron publicados en revistas académicas entre 2007 hasta el 2018 [160]. Los resultados analíticos revelaron que los lenguajes basados en bloques son los más utilizados en el proceso. Se podría decir que los lenguajes visuales son una herramienta para introducir el aprendizaje de la programación textual, desarrollando habilidades como la creatividad, habilidades de resolución de problemas y desarrollo del pensamiento computacional.

A través de la revisión de la literatura sobre el uso del entorno ALICE como herramienta para el aprendizaje de la programación, se reveló la eficacia del software ALICE como recurso de soporte para este proceso, además de que incide de manera positiva en el rendimiento de los estudiantes en la mayoría de los trabajos analizados. De igual manera, se evidencia información sobre experiencias de inclusión de participantes, como es el caso de estudiantes de género femenino, o grupos culturales, entre otros colectivos.

Inicialmente, se presenta ALICE como un lenguaje de programación innovador, que favorece la creación de animaciones para narrar historias y la creación de juegos para difundir en la web; planteando, además, que es una herramienta de enseñanza para la introducción a la programación, básicamente por la facilidad de uso para principiantes debido a su interfaz de arrastrar y soltar elementos, y su amplia biblioteca de objetos 3D.

Con el propósito de conocer las experiencias de otros entornos de programación diferentes a ALICE, se realizó una revisión de literatura sobre la enseñanza de programación usando el MIT App Inventor [162], un lenguaje de programación visual basado en bloques, ya que es muy utilizado a nivel general para aprender fundamentos de programación. Se fundamenta en la programación móvil para programadores novatos. Los resultados evidencian que MIT App Inventor goza de una aceptación en la comunidad académica como un entorno eficaz para desarrollar la motivación e incrementar el rendimiento de los participantes que se inician en programación, de cualquier nivel educativo,

lo que constituye un referente importante para la evaluación de ALICE como recurso alternativo.

Con las revisiones de literatura realizada, se evalúa el uso de ALICE como programa educativo para el aprendizaje de la programación en estudiantes del primer año de la carrera de ingeniería en software de la Universidad Estatal de Milagro, Milagro-Ecuador. Para evaluar ALICE se realizó un experimento controlado. Mediante la adopción de la Revisión Sistemática de Literatura y el Experimento como métodos de investigación, se logra el objetivo general de evaluar ALICE en el contexto educativo planteado.

En cuanto a la experiencia de la aplicación ALICE, entre otras plataformas de aprendizaje, se demuestra la efectividad del entorno ALICE para apoyar el proceso de aprendizaje de programación en estudiantes del curso introductorio a la programación en la Universidad Estatal de Milagro, Ecuador. Esta conclusión se basa en el diseño de dos grupos de sujetos: control, con el enfoque tradicional; y experimental, con el entorno ALICE; cuyos resultados, un promedio de notas de 80 en el grupo experimental y 65 en el grupo de control, ponen en evidencia la ventaja de utilizar ALICE.

En particular, a partir de los resultados de la experiencia en estudiantes de Ingeniería en software de la Universidad Estatal de Milagro, se evidencia que existe una reducción de fracasos en el aprendizaje de la programación, comparando los promedios de notas entre el primero y el segundo parcial existiendo un incremento en las notas después de que los estudiantes aprendieran programación con ALICE en el segundo parcial. Por otro lado, no se reportaron diferencias significativas al comparar los promedios de notas entre los grupos en función del género (femenino, masculino), ni el turno (matutino y nocturno); sin embargo, se evidencia la influencia positiva que tiene la asistencia regular a clases sobre los resultados del desempeño académico, y mejores resultados en el desempeño académico en el grupo experimental que participó en el taller ALICE con respecto a los que aprendieron programación de la manera tradicional.

8.2. Debate, limitaciones y extensiones

En este apartado se exponen las principales decisiones que se adoptaron en la presente tesis, destacando sus principales limitaciones y posibles extensiones.

- **¿Es apropiada una revisión de la literatura para establecer la evidencia empírica de la aplicación de los lenguajes de programación en un contexto educativo?**

  Con la revisión de la literatura realizada sobre los programas educativos,

se identificó la necesidad de mejores enfoques con evidencia empírica en los resultados de investigaciones sobre el uso de lenguajes de programación con fines de aprendizaje. Así mismo, se identificaron algunas oportunidades de investigación relacionadas con los lenguajes de programación utilizados, las áreas o etapas de su aplicación, la necesidad de más evidencia en general, y más estudios en contexto non WEIRD (occidentales, educados, industrializados, ricos y democráticos). Para esta revisión, se analizaron diferentes métodos y herramientas en diferentes niveles educativos y con diferentes objetivos, a partir de veintinueve (29) estudios que presentaban alguna evidencia empírica, de un total de sesenta y dos (62) preseleccionados.

- *Conclusión:* la RSL que se realizó devolvió información valiosa para la presente tesis, por tanto es una opción apropiada para obtener información sobre la evidencia empírica.
- *Extensión:* revisar el proceso, actualizar las estrategias de búsqueda, incluyendo los criterios de inclusión y exclusión.

- **¿Se puede considerar oportuna una revisión sistemática de la literatura para determinar la efectividad del software ALICE?**

  A partir de los hallazgos de la revisión sistemática de literatura se pone en evidencia, por un lado, la eficacia del uso de ALICE como recurso de aprendizaje el proceso introductorio a la programación; y por otro, la escasa validez de los resultados experimentales. Esta primera conclusión, se basa en el análisis de veinticuatro (24) artículos seleccionados, siguiendo un protocolo de búsqueda y criterios preestablecidos, de las bases de datos más comunes en el área de ciencias de la computación, como ACM, IEEE, entre otras.

  En general, con la base en la revisión de los trabajos relacionados, la investigación en aprendizaje de la programación en ciencias de la computación es un área emergente, donde las diferentes experiencias con el entorno ALICE han dejado en evidencia su eficacia como recurso de apoyo. Sin embargo, se necesita de nuevas iniciativas a objeto de validar en otros contextos, así como la evaluación de las causas que pudieran impactar en la determinación de la eficacia.

  - *Conclusión:* Para esta primera RSL, se obtuvo información general de los hallazgos de cada trabajo analizado. Si se tuviese que decidir si realizar una RSL o no, se volvería a realizar para la presente tesis.
  - *Extensión:* nueva revisión de literatura para identificar nuevas dimensiones e información mas detallada en la literatura de la aplicación de ALICE en el proceso de aprendizaje de la programación.

- **¿Con la realización de un experimento en el contexto educativo se puede establecer la incidencia del software ALICE en el aprendizaje de la programación?**

  Aunque las propuestas de experiencias con el entorno ALICE en el aprendizaje introductorio de programación reportan la efectividad de esta plataforma como una herramienta de apoyo en este proceso, se destaca el poco interés en la validez de los resultados obtenidos en la mayoría de los estudios, por lo que se sugiere sustentar sobre este tema en futuras investigaciones. En este sentido, sería importante crear una línea de investigación con enfoque experimental con la cual se genere evidencia empírica, con base en los estudios críticos enfocados en orientar y promover la investigación en este campo. Como ya se ha planteado en el marco de esta tesis, el informar sobre estudios con una estrategia empírica definida puede ayudar a los profesionales a mejorar el uso de los lenguajes de programación en el proceso educativo.

  - *Conclusión:* si bien es cierto, el experimento se realizó de acuerdo a lo planificado, pero con ciertas limitaciones en cuanto a la selección de los estudiantes que participaron en el mismo.
  - *Extensión:* replicar el experimento para comparar los resultados y comprobar la efectividad del software ALICE.

Por otra parte, aunque se presentan experiencias en temas de inclusión de estudiantes (como, por ejemplo, estudiantes de género femenino, grupos culturales, entre otros colectivos), se evidencian pocos trabajos en Latinoamérica, donde se encontraron experiencias aisladas en México y Colombia. Se recomienda desarrollar un proyecto interuniversitario para el contexto lationamericano, dada la subrepresentación que tiene esta región en los trabajos revisados. Se podrían utilizar las redes interuniversitarias para subsanar la falta de estudios en contextos non-WIERD, y enfrentar el desafío que tienen estudiantes del sector rural con la baja preparación académica, aislamiento geográfico y malas condiciones socioeconómicas. Así mismo, un proyecto interuniversitario podría suavizar el sesgo de realizar únicamente estudios en países similares.

De igual manera, se recomienda promover investigaciones que incentiven el aprendizaje de programación en educación temprana. En esta tesis se pone en evidencia que existen pocos estudios que muestren los factores que expliquen el por qué los lenguajes tradicionales no son adecuados para aprender a programar, así como la necesidad de comprender la dificultades que tienen los estudiantes y cómo hacer para superar aquellas en las etapas tempranas del proceso educativo.

Sería importante para los investigadores en el área el diseño de una meto-

dología que permita la comparación entre las diferentes propuestas. Esto les permitiría la discusión y mejora de las mismas, con criterios e indicadores de logro, de forma que se pudiera tener un mejor panorama de la eficacia de las acciones para la mejora de la enseñanza de programación.

Finalmente, y dado que no se tiene resultado concluyente sobre el uso de diferentes plataformas para aprender a programar basándose en lenguajes basados en bloques, se requiere la promoción de la investigación sobre el tema.

Parte V

Apéndice

Apéndice
Cadenas de Búsqueda

Esta sección contiene las cadenas de búsqueda aplicadas en las base de datos que se usaron para obtener información de los estudios con evidencia empírica, realizado en la Sección §6.2.1 del Capítulo §6.

Base de datos	Cadena de búsqueda	Resultados
SCOPUS	(TITLE-ABS-KEY(Learning programming OR Learning coding OR Learning to program OR Learning to code OR Learning of computer programming) AND TITLE-ABS-KEY(Educational software OR Educational technology OR Educational programming language OR Visual programming environment OR Block-based languages OR Block-based coding)) AND PUBYEAR >2006 AND (LIMIT-TO(SUBJAREA, ÇOMP") OR LIMIT-TO(SUBJAREA, "SOCI")) AND (LIMIT-TO(LANGUAGE, ``English''))	39
ACM	+("Learning programmingLearning codingLearning to programLearning to codeLearning of computer programming")+(.Educational software Educational technologyEducational programming languageVisual programming environmentBlock-based languagesBlock-based coding")	36
IEEE	("Learning programming.°R "Learning coding.°R "Learning to program.°R "Learning to code.°R "Learning of computer programming") AND (.Educational software.°R .Educational technology.°R .Educational programming language.°R "Visual programming environment.°R "Block-based languages.°R "Block-based coding")	12
SCIENCE DIRECT	("Learning programming.°R "Learning coding.°R "Learning to program.°R "Learning to code.°R "Learning of computer programming") AND (.Educational software.°R .Educational technology.°R .Educational programming language.°R "Visual programming environment.°R "Block-based languages.°R "Block-based coding")	438
WILEY	("Learning programming.°R "Learning to program.°R "Learning of computer programming") AND (.Educational software.°R .Educational technology.°R .Educational programming language.°R "Visual programming environment.°R "Block-based languages.°R "Block-based coding")	374

Tabla 8.1: Cadenas de búsquedas de las bases de datos

Bibliografía

[1] S. A-Ghamdi, N. Al-Rajhi, N. Al-Onaizy y H. Al-Khalifa. Using app inventor 2 in a summer programming workshop: Improvements over previous years. páginas 383–388, 04 2016

[2] J. Adell, M. A. Llopis, F. Esteve y G. Valdeolivas. El debate sobre el pensamiento computacional en educación. *RIED. Revista Iberoamericana de Educación a Distancia*, 22:171–186, 12 2018

[3] A. V. Aho. Computation and computational thinking. *Comput. J.*, 55(7): 832–835, julio 2012

[4] S. Al-Sabbagh, H. Gedawy, H. Alshikhabobakr y S. Razak. Computing curriculum in middle schools: An experience report. En *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '17, página 230–235, New York, NY, USA, 2017. Association for Computing Machinery

[5] K. Al-Tahat. An innovative instructional method for teaching object-oriented modelling. *International Arab Journal of Information Technology*, 11(6), 2014

[6] K. Al-Tahat. Alice adventures in computingland: A review. En *2021 22nd International Arab Conference on Information Technology (ACIT)*, páginas 1–7, 2021

[7] K. Al-Tahat, N. Taha, B. Hasan y B. Shawar. The impact of a 3d visual tool on female students attitude and performance in computer programming. páginas 864–867, 07 2016

[8] A. Ali y D. Smith. Teaching an introductory programming language in a general education course. *Journal of Information Technology Education: Innovations in Practice*, 13:057–067, 01 2014

[9] B. Alshaigy, S. Kamal, F. Mitchell, C. Martin y A. Aldea. PILeT: An interactive learning tool to teach python. En *WiPSCE*, volumen 09-11-Nove, páginas 76–79, 2015

[10] C. Angeli, J. Voogt, A. Fluck, M. Webb, M. Cox, J. Malyn-Smith y J. Zagami. A k-6 computational thinking curriculum framework: Implications for teacher knowledge. 19:47–57, 01 2016

[11] C. Angevine, K. Cator, J. Roschelle y S. A. Thomas. Computational thinking for a computational world. Technical report, 2017

[12] G. Aranda y J. P. Ferguson. Unplugged programming: The future of teaching computational thinking? *Pedagogika*, 68, 12 2018

[13] T. C. Armstrong y R. F. Loane. Educational software: A developer's perspective. *TechTrends*, 39(1):20–22, Jan 1994.

[14] K. Assiter y C. Wiseman. Exploratory learning with alice : Experiences leading a computer science workshop for girl scouts. *Journal of Computing Sciences in Colleges*, 31:21 – 27, 04 2016

[15] J. Aycock, E. Pitout y S. Storteboom. A Game Engine in Pure Python for CS1: Design, Experience, and Limits. En *ITiCSE*, ITiCSE '15, páginas 93–98, New York, NY, USA, 2015. ACM

[16] M. A. Babar y H. Zhang. Systematic literature reviews in software engineering: Preliminary results from interviews with researchers. En *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, páginas 346–355, 2009

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

[18] L. C. Begosso, L. R. Begosso y N. A. Christ. An analysis of block-based programming environments for cs1. volumen 2020-October, 2020.

[19] T. Bell, J. Alexander, I. Freeman y M. Grimley. Computer science unplugged: school students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, 13, 01 2009

[20] M. Ben-Ari, R. Bednarik, R. B.-B. Levy, G. Ebel, A. Moreno, N. Myller y E. Sutinen. A decade of research and development on program animation: The Jeliot experience. *JVLC*, 22(5):375–384, 2011

[21] N. Besshaposhnikov, A. Kushnirenko y A. Leonov. Pictomir: how and why do we teach textless programming for preschoolers, first graders and students of pedagogical universities. páginas 1–7, 10 2017

[22] S. M. Biju. Taking advantage of alice to teach programming concepts. *E-Learning and Digital Media*, 10(1):22–29, 2013

[23] B. S. Bloom, M. B. Engelhart, E. J. Furst, W. H. Hill y D. R. Krathwohl. *Taxonomy of educational objectives. The classification of educational goals. Handbook 1: Cognitive domain*. Longmans Green, New York, 1956

[24] S. Bocconi, A. Chioccariello, G. Dettori, A. Ferrari y K. Engelhardt. *Developing Computational Thinking in Compulsory Education: Implications for policy and practice*. Número June. Luxembourg: Publications Office of the European Union, 2016

[25] M. Bonilla-del Río y I. Aguaded. La escuela en la era digital: smartphones, apps y programación en educación primaria y su repercusión en la competencia mediática del alumnado. *Píxel-Bit. Revista de Medios y Educación*, 0(53):151–163, 2018

[26] Y. Bosse y M. A. Gerosa. Why is programming so difficult to learn?: Patterns of difficulties related to programming learning mid-stage. *ACM SIGSOFT Software Engineering Notes*, 41:1–6, 01 2017

[27] N. Brown, J. Monig, A. Bau y D. Weintrop. Future directions of block-based programming. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, página 315–316, 02 2016

[28] F. Buitrago, R. Casallas, M. Hernandez Hoyos, A. Reyes, S. Restrepo y G. Danies. Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, 87:834–860, 08 2017

[29] J. Cárdenas-Cobo, A. Puris, P. Novoa-Hernández, Á. Parra-Jiménez, J. Moreno-León y D. Benavides. Using scratch to improve learning programming in college students: A positive experience from a non-weird country. *Electronics (Switzerland)*, 10(10), 2021.

[30] J. Carrión-Martínez, A. Rosa, J. Fernández y M. Montenegro-Rueda. Information and communications technologies (icts) in education for sustainable development: A bibliographic review. *Sustainability*, 12:3288, 04 2020

[31] A. S. Carter, C. D. Hundhausen y O. Adesope. Blending measures of programming and social behavior into predictive models of student achievement in early computing courses. *ACM Trans. Comput. Educ.*, 17(3):12:1–12:20, agosto 2017

[32] I. Cetin y E. Dubinsky. Reflective abstraction in computational thinking. *The Journal of Mathematical Behavior*, 47:70 – 80, 2017

[33] C.-K. Chang. Effects of using alice and scratch in an introductory programming course for corrective instruction. *Journal of Educational Computing Research*, 51:185–204, 09 2014

[34] Z. Chang, Y. Sun, T.-Y. Wu y M. Guizani. Scratch analysis tool(sat): A modern scratch project analysis tool based on antlr to assess computational thinking skills. páginas 950–955, 06 2018

[35] P.-Y. Chao. Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *C&E*, 95:202–215, 2016

[36] P. Chen, Y. Tian, W. Zhou y R. Huang. A systematic review of computational thinking: Analysing research hot spots and trends by CiteSpace. *ICCE 2018 - 26th International Conference on Computers in Education, Main Conference Proceedings*, páginas 211–213, 2018

[37] S. Cohen, R. Chechile, G. Smith, F. Tsai y G. Burns. A method for evaluating the effectiveness of educational software. *Behavior Research Methods, Instruments, & Computers*, 26(2):236–241, Jun 1994

[38] S. Coomans y G. Santos Lacerda. Petese, a pedagogical ergonomic tool for educational software evaluation. *Procedia Manufacturing*, 3:5881 – 5888, 2015.

[39] S. Cooper, W. Dann y R. Pausch. Alice: A 3-d tool for introductory programming concepts. *Journal of Computing Sciences in Colleges - JCSC*, 15, 01 2000

[40] S. Cooper, W. Dann y R. Pausch. Teaching objects-first in introductory computer science. *SIGCSE Bull.*, 35(1):191–195, jan 2003

[41] J. Costa. Using alice software with the expository method: A pilot study. *International Journal of Engineering and Technology*, 10:1681–1686, 12 2018

[42] J. M. Costa y G. L. Miranda. Relation between Alice software and programming learning: a systematic review of the literature and meta-analysis. *BJET*, páginas n/a—-n/a, 2016

[43] M. Ćurčić, D. Milinković y D. Radivojević. Educational computer software in the function of integrating and individualization in teaching of mathematics and knowledge of nature. *Eurasia Journal of Mathematics, Science and Technology Education*, 14, 07 2018

[44] T. Daly. *Influence of Alice 3: Reducing the Hurdles to Success in a CS1 Programming Course*.

https://digital.library.unt.edu/ark:/67531/metadc271795/, University of North Texas, May 2013

[45] G. Dambić, T. Keŝĉec y D. Kuĉak. A blended learning with gamification approach for teaching programming courses in higher education. En *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, páginas 843–847, 2021

[46] N. Davis. *Digital technologies and change in education: The arena framework*. 01 2017

[47] J. Denner, L. Werner, S. Campe y E. Ortiz. Using game mechanics to measure what students learn from programming games. *International Journal of Game-Based Learning*, 4:13–22, 07 2014

[48] P. J. Denning. Remaining trouble spots with computational thinking. *Commun. ACM*, 60(6):33–39, mayo 2017

[49] N. Z. Dina, E. Wuryanto y R. S. Marjianto. Evaluation on the effectiveness of visual learning environment on programming course from students' perspectives. *IIUM Engineering Journal*, 20(1):100–107, 2019.

[50] J. Dwarika. *The use of ALICE, a visual environment for teaching and learning object-oriented programming*. Tesis doctoral, University of South Africa, 2014

[51] J. Dwarika y R. De Villiers. Use of the alice visual environment in teaching and learning object-oriented programming. En *ACM International Conference Proceeding Series*, volumen 28-30-September-2015, 2015.

[52] H. K. Edwards, J. L. Gersting y T. Tangaro. Teaching alice in hawai'i: Cultural perspectives. En *2007 37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports*, páginas T3A–1–T3A–5, 2007

[53] M. Endalew y G. Tessema. Role of information and communication technologies in educational systems: a systematic review. *International Journal of Scientific Reports*, 6:277–282, 06 2020

[54] O. Erol y A. A. Kurt. The effects of teaching programming with scratch on pre-service information technology teachers' motivation and achievement. *Computers in Human Behavior*, 77:11 – 18, 2017

[55] M. Esteves, B. Fonseca, L. Morgado y P. Martins. Improving teaching and learning of computer programming through the use of the second life virtual world. *British Journal of Educational Technology*, 42(4):624–637, 2011

[56] J. P. Febles R. y A. Gonzalez-Perez. Aplicación de la minería de datos en la bioinformática. *Revista Cubana de Información en Ciencias de la Salud*, 04 2002

[57] A. Feng, M. Gardner y W.-c. Feng. Parallel programming with pictures is a snap! *Journal of Parallel and Distributed Computing*, 105:150–162, 2017.

[58] C.-Y. Feng y M.-P. Chen. The effects of goal specificity and scaffolding on programming performance and self-regulation in game design. *BJET*, 45 (2):285–302, 2014

[59] B. Ferede, J. Elen, W. van Petegem, A. B. Hunde y K. Goeman. Instructors' educational ict use in higher education in developing countries: evidence from three ethiopian universities. *Journal of Computing in Higher Education*, 2022

[60] J. M. Fernández, M. E. Zúñiga, M. V. Rosas y R. A. Guerrero. Experiences in Learning Problem-Solving through Computational Thinking. *Journal of Computer Science and Technology*, 18(02):e15, 2018

[61] P. Ferreira, L. Nogueira, N. Pereira, C. Maia, M. Fernandes, A. Andrade, R. Faria y C. Gonçalves. Teaching programming with a limited infrastructure. En *2021 World Engineering Education Forum/Global Engineering Deans Council (WEEF/GEDC)*, páginas 556–562, 2021

[62] J. Fuentes-Rosado y M. Moo Medina. Dificultades de aprender a programar. *Revista Educación en Ingeniería*, 12:76, 07 2017

[63] A. García-Valcárcel Muñoz-Repiso y Y.-A. Caballero-González. Robótica para desarrollar el pensamiento computacional en Educación Infantil. *Comunicar*, 59(XXVII):63–72, 2019

[64] D. Goulet y D. Slater. Alice and the introductory programming course: An invitation to dialogue. *Number*, 7, 07 2009

[65] E. Graczyńska. Alice as a tool for programming at schools. *Natural Science*, 02:124–129, 01 2010

[66] S. Grover, S. Basu, M. Bienkowski, M. Eagle, N. Diana y J. Stamper. A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based programming environments. *ACM Trans. Comput. Educ.*, 17(3), aug 2017

[67] S. Grover y R. Pea. Computational thinking in k–12 a review of the state of the field. *Educational Researcher*, 42:38–43, 02 2013

[68] R. Harimurti, E. Ekohariadi, M. Munoto y A. Buditjahjanto. The concept of computational thinking toward information and communication technology learning. *IOP Conference Series: Materials Science and Engineering*, 535:012004, 06 2019

[69] R. Harimurti, A. Qoiriah, E. Ekohariadi y M. Munoto. Implementation of computational thinking concepts in ict learning using scratch programming. En *International Conference on Indonesian Technical Vocational Education and Association (APTEKINDO 2018)*. Atlantis Press, 2018/07

[70] J. Harrison. Alice in virginia beach, a continuing experiment. En *Proceedings of Alice Symposium on Alice Symposium*, ALICE '13, New York, NY, USA, 2013. Association for Computing Machinery

[71] H. Haseski, U. Ilic y U. Tugtekin. Defining a new 21st century skill-computational thinking: Concepts and trends. *International Education Studies*, 11:29, 03 2018

[72] R. Horváth y S. Javorský. New Teaching Model for Java Programming Subjects. En *WCES*, volumen 116, páginas 5188–5193, 2014

[73] J. Huber y N. B. Giuse. Educational software evaluation process. *Journal of the American Medical Informatics Association*, 2(5):295–296, 1995

[74] F. Huch. Learning Programming with Erlang. En *ERLANG*, ERLANG '07, páginas 93–99, New York, NY, USA, 2007. ACM

[75] U. Ilic, H. Haseski y U. Tugtekin. Publication trends over 10 years of computational thinking research. *Contemporary Educational Technology*, 9, 04 2018

[76] A. Ioannidou, A. Repenning y D. Webb. Agentcubes: Incremental 3d end-user development. *J. Vis. Lang. Comput.*, 20:236–251, 08 2009

[77] O. Iskrenovic-Momcilovic. Learning a programming language. *International Journal of Electrical Engineering Education*, 55:324–333, 05 2018

[78] S. Janpla y P. Piriyasurawong. The development of problem-based learning and concept mapping using a block-based programming model to enhance the programming competency of undergraduate students in computer science. *TEM Journal*, 7(4):708–716, 2018

[79] K. Johnsgard y J. McDonald. Using alice in overview courses to improve success rates in programming i. páginas 129 – 136, 05 2008

[80] M. E. M. Jones, M. Kisthardt y M. A. Cooper. Interdisciplinary teaching: Introductory programming via creative writing. En *Proceedings*

of the 42nd ACM Technical Symposium on Computer Science Education, SIGCSE '11, página 523–528, New York, NY, USA, 2011. Association for Computing Machinery

[81] R. Kadar, N. A. Wahab, J. Othman, M. Shamsuddin y S. B. Mahlan. A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review. *International Journal of Academic Research in Progressive Education and Development*, 10(3):591—-605, 2021

[82] Y. Kafai y V. Vasudevan. Hi-Lo Tech Games: Crafting, Coding and Collaboration of Augmented Board Games by High School Youth. En *IDC '15*, páginas 130–139, New York, NY, USA, 2015. ACM

[83] C. M. Kandemir, F. Kalelioglu y Y. Gulbahar. Pedagogy of teaching introductory text-based programming in terms of computational thinking concepts and practices. *Computer Applications in Engineering Education*, 29(1):29–45, 2021.

[84] H. Kang, J. Cho y H. Kim. Application study on android application prototyping method using app inventor. *Indian Journal of Science and Technology*, 8(19), 2015

[85] N. Karaman Aksentijević, Z. Jezić y P. A. Zaninović. The effects of information and communication technology (ict) use on human development—a macroeconomic approach. *Economies*, 9(3), 2021

[86] C. Kelleher y R. Pausch. Using storytelling to motivate programming. *Commun. ACM*, 50:58–64, 07 2007

[87] B. A. Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, 33, 08 2004

[88] B. A. Kitchenham, D. Budgen y P. Brereton. *Evidence-Based Software Engineering and Systematic Reviews*. Chapman & Hall/CRC, 2015

[89] A. J. Ko y B. A. Myers. A framework and methodology for studying the causes of software errors in programming systems. *Journal of Visual Languages Computing*, 16(1):41 – 84, 2005.

[90] M. Kordaki. A drawing and multi-representational computer environment for beginners' learning of programming using C: Design and pilot formative evaluation. *C&E*, 54(1):69–87, 2010

[91] M. Kosa, M. Yilmaz, R. O'Connor y P. Clarke. Software engineering education and games: A systematic literature review. *Journal of Universal Computer Science*, 22:1558–1574, 12 2016

[92] B. Kules. Computational thinking is critical thinking: Connecting to university discourse, goals, and learning outcomes. *Proceedings of the Association for Information Science and Technology*, 53(1):1–6, 2016

[93] I. Lee, F. Martin, J. Denner, B. Coulter, W. Allan, J. Erickson, J. Malyn-Smith y L. Werner. Computational thinking for youth in practice. *ACM Inroads*, 2:32–37, 02 2011

[94] M. J. Lee y A. J. Ko. Comparing the Effectiveness of Online Learning Approaches on CS1 Learning Outcomes. En *ICER*, ICER '15, páginas 237–246, New York, NY, USA, 2015. ACM

[95] J. Leonard, M. Mitchell, J. Barnes-Johnson, A. Unertl, J. Outka-Hill, R. Robinson y C. Hester-Croff. Preparing teachers to engage rural students in computational thinking through robotics, game design, and culturally responsive teaching. *Journal of Teacher Education*, 69: 002248711773231, 09 2017

[96] J. Leskovec y R. Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Trans. Intell. Syst. Technol.*, 8(1):1:1–1:20, julio 2016

[97] Y. Li. Teaching programming based on computational thinking. En *2016 IEEE Frontiers in Education Conference (FIE)*, páginas 1–7, Oct 2016

[98] M. H. Lino Ferreira da Silva Barros, A. Rodrigues y E. C. Dos Santos Silva. A systematic literature review for multimedia learning objects applied to stewart platforms using software engineering methods. 7:1–15, 01 2019

[99] S. Y. Lye y J. H. Ling Koh. Review on teaching and learning of computational thinking through programming: What is next for k-12? *Computers in Human Behavior*, 41:51 – 61, 2014

[100] M. Mac Gaul de Jorge, M. L. Massé Palermo y N. Sarmiento Barbieri. Análisis de alice para la enseñanza básica de la programación. En *VIII Congreso de Tecnología en Educación y Educación en Tecnología*, 2013

[101] R. P. Medeiros, G. L. Ramalho y T. P. Falcão. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, páginas 1–14, 2019

[102] A. Merkouris y K. Chorianopoulos. Introducing Computer Programming to Children Through Robotic and Wearable Devices. En *WiPSCE*, WiPSCE '15, páginas 69–72, New York, NY, USA, 2015. ACM

[103] A. Merkouris, K. Chorianopoulos y A. Kameas. Teaching programming in secondary education through embodied computing platforms: Robotics and wearables. *TCE*, 17(2):9:1–9:22, mayo 2017

[104] D. Midian Kurland, R. Pea, C. Clement y R. Mawby. A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research*, 2, 01 1986

[105] E. Milková. Multimedia application for educational purposes: Development of algorithmic thinking. *Applied Computing and Informatics*, 11, 05 2014

[106] S. Minör. Interacting with structure-oriented editors. *International Journal of Man-Machine Studies*, 37(4):399 – 418, 1992.

[107] L. Morales Diaz, L. S. Gaytan-Lugo y L. Fleck. Profiling styles of use in alice: Identifying patterns of use by observing participants in workshops with alice. En *Proceedings of the 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, BLOCKS AND BEYOND '15, página 19–24, USA, 2015. IEEE Computer Society

[108] J. Moreno-León. *On the Development of Computational Thinking Skills in Schools through Computer Programming with Scratch*. Tesis doctoral, Escuela Internacional de Doctorado, 2018

[109] J. Moreno-León y G. Robles. Code to learn with scratch? a systematic literature review. En *EDUCON 2016*, páginas 150–156, April 2016

[110] P. Mullins, D. Whitfield y M. Conlon. Using alice 2.0 as a first language. *Journal of Computing Sciences in Colleges*, 24:136–143, 01 2009

[111] E. Nardelli. Do we really need computational thinking? *Commun. ACM*, 62(2):32–35, enero 2019

[112] C. C. Navarrete. Creative thinking in digital game design and development: A case study. *C&E*, 69:320–331, 2013

[113] S. A. Nikou y A. A. Economides. Transition in student motivation during a scratch and an app inventor course. En *EDUCON*, páginas 1042–1045, 2014

[114] M. Noone y A. Mooney. Visual and textual programming languages: A systematic review of the literature. *CoRR*, abs/1710.01547, 2017

[115] J. Noshin y S. Ahmed. Teaching programming to non-programmers at undergraduate level. *International Journal of Engineering and Management Research*, 8, 03 2018

[116] A. Ortega García, A. Ruiz-Martínez y R. Valencia-García. Using app inventor for creating apps to support m-learning experiences: A case study. *CAEE*, 26(3):431–448, 2018

[117] S. Papadakis y V. Orfanakis. The combined use of lego mindstorms nxt and app inventor for teaching novice programmers. *AISC*, 560:193–204, 2017

[118] S. Parra Sarmiento, M. Gómez Zermeño y M. Pintor Chávez. Factores que inciden en la implementación de las tic en los procesos de enseñanza-aprendizaje en 5º de primaria en colombia. *Revista Complutense de Educación*, 26:197–213, feb. 2015

[119] G. Pellone. Educational software design: A literature review. *Australasian Journal of Educational Technology*, 11(1), 1995.

[120] G. Petri y C. Gresse von Wangenheim. How games for computing education are evaluated? a systematic literature review. *Computers Education*, 107:68 – 90, 2017

[121] S. Psycharis y E. Kotzampasaki. *Eurasia Journal of Mathematics, Science and Technology Education*, 15(4), 2019

[122] J. Pérez, J. Díaz, J. Garcia-Martin y B. Tabuenca. Systematic literature reviews in software engineering—enhancement of the study selection process using cohen's kappa statistic. *Journal of Systems and Software*, 168:110657, 2020

[123] F. Rahman. Leveraging visual programming language and collaborative learning to broaden participation in computer science. En *SIGITE*, páginas 172–177, 2018.

[124] S. Reardon y B. Tangney. Smartphones, Studio-Based Learning, and Scaffolding: Helping Novices Learn to Program. *TCE*, 14(4):23:1—-23:15, 2014

[125] A. Repenning, A. R. Basawapatna y N. A. Escherle. *Principles of Computational Thinking Tools*, páginas 291–305. Springer International Publishing, Cham, 2017

[126] A. Repenning y T. Sumner. Agentsheets: a tool for building visual programming environments. páginas 30–30, 01 1992

[127] A. Rizzo, F. Montefoschi, S. Ermini y G. Burresi. Udoo app inventor: Introducing novices to the internet of things. *International Journal of People-Oriented Programming*, 4:33–49, 01 2015

[128] S. H. Rodger, M. Bashford, L. Dyck, J. Hayes, L. Liang, D. Nelson y H. Qin. Enhancing k-12 education with alice programming adventures.

En *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '10, página 234–238, New York, NY, USA, 2010. Association for Computing Machinery

[129] I. Rogozhkina y A. Kushnirenko. PiktoMir: teaching programming concepts to preschoolers with a new tutorial environment. En *WCETR*, volumen 28, páginas 601–605, 2011

[130] A. Rojas López y F. J. García-Peñalvo. Relationship of knowledge to learn in programming methodology and evaluation of computational thinking. En *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM '16, páginas 73–77, New York, NY, USA, 2016. ACM

[131] J. Roschelle, C. DiGiano, M. Koutlis, A. Repenning, J. Phillips, N. Jackiw y D. Suthers. Developing educational software components. *Computer*, 32(9):50–58, Sep. 1999

[132] N. F. Rozali y N. M. Zaid. Code puzzle: Actionscript 2.0 learning application based on problem based learning approach. En *ISPC*, volumen 2017-January, páginas 1–4, 2017.

[133] P. Runeson, M. Höst, A. Rainer y B. Regnell. *Case Study Research in Software Engineering – Guidelines and Examples*. 02 2012

[134] M. Saeli, J. Perrenet, W. Jochems y B. Zwaneveld. Teaching programming in secondary school: A pedagogical content knowledge perspective. *Informatics in Education*, 10:73–88, 04 2011

[135] J.-M. Sáez-López y R. Cózar. Pensamiento computacional y programación visual por bloques en el aula de primaria. *Educar*, 53:129–146, 12 2016

[136] S. M. Salleh, Z. Shukur y H. M. Judi. Analysis of research in programming teaching tools: An initial review. *Procedia - Social and Behavioral Sciences*, 103:127 – 135, 2013

[137] A. Sattar y T. Lorenzen. Teach alice programming to non-majors. *SIGCSE Bull.*, 41(2):118–121, jun 2009

[138] L. Schultz. Student perceptions of instructional tools in programming logic: A comparison of traditional versus alice teaching environments. *Information Systems Education Journal*, 9:60–66, 2011

[139] E. Segredo, G. Miranda Valladares y C. Leon. Towards the education of the future: Computational thinking as a generative learning mechanism. *Education in the Knowledge Society (EKS)*, 18:33, 07 2017

[140] C. Selby. Relationships: computational thinking, pedagogy of programming, and bloom's taxonomy. En *Proceedings of the 10thWorkshop in Primary and Secondary Computing Education (WiPSCE'15)*, 01 2015

[141] C. Selby. Four approaches to teaching programming. página 10, 12 2021

[142] C. Selby y J. Woollard. Computational Thinking : The Developing Definition. *ITiCSE Conference 2013*, páginas 5–8, 2013

[143] C. Selby y J. Woollard. Refining an understanding of computational thinking. *University of Southampton, UK*, 12 2014

[144] R. B. Shapiro y M. Ahrens. Beyond blocks: Syntax and semantics. *Commun. ACM*, 59(5):39–41, abril 2016

[145] S. D. Sharma. *Library automation software packages used in academic libraries of Nepal: a comparative study*. Tesis doctoral, National Institute of Science Communication, 2007

[146] V. Sharma. A literature review on an effective use of ict in education. *Journal of Computer Science and Technology Studies*, 2(1):9–17, Jun. 2020

[147] D. Smith, P. Lameras y N. Moumoutzis. Using educational programming languages to enhance teaching in computer science. 07 2019

[148] H.-J. So y M. Seo. *A systematic literature review of game-based learning and gamification research in Asia*, capítulo A systematic literature review of game-based learning and gamification research in Asia, páginas 396–413. Routledge, 2018

[149] D. Stanojević, D. Cenić y S. Cenić. Application of computers in modernization of teaching science. *International Journal of Cognitive Research in Science Engineering and Education*, 6:89–104, 01 2018

[150] L. Sun, L. Hu y D. Zhou. Programming attitudes predict computational thinking: Analysis of differences in gender and programming experience. *Computers Education*, 181:104457, 2022

[151] E. Tanrikulu y B. C. Schaefer. The users who touched the ceiling of scratch. En *WCETR*, volumen 28, páginas 764–769, 2011

[152] M. Tedre y P. J. Denning. The long quest for computational thinking. En *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, Koli Calling '16, páginas 120–129, New York, NY, USA, 2016. ACM

[153] M. Tempel. Blocks programming. *CSTA Voice*, 9(1):1–3, 2013

[154] The British Royal Society. Shut down or restart? *British Journal of Educational Technology*, (January):789–801, 2012

[155] H. Ting-Chia, C. Shao-Chen y H. Yu-Ting. How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, 126:296 – 310, 2018

[156] G. Tuparov, D. Tuparova y V. Jordanov. Teaching Sorting and Searching Algorithms through Simulation-based Learning Objects in an Introductory Programming Course. En *WCES*, volumen 116, páginas 2962–2966, 2014

[157] H. Ulrich Hoppe y S. Werneburg. *Computational Thinking—More Than a Variant of Scientific Inquiry!*, páginas 13–30. Springer Singapore, Singapore, 2019

[158] I. Utting, S. Cooper, M. Kölling, J. Maloney y M. Resnick. Alice, greenfoot, and scratch - a discussion. *ACM Transactions on Computing Education (TOCE)*, 10:17, 11 2010

[159] J. Valverde-Berrocoso, M. Fernández Sánchez y M. Arroyo. El pensamiento computacional y las nuevas ecologías del aprendizaje. *RED - Revista de Educación a Distancia*, páginas 1–18, 09 2015

[160] M. Vinueza-Morales, D. Borrego, J. A. Galindo y D. Benavides. Empirical evidence of the usage of programming languages in the educational process. *IEEE Transactions on Education*, 64(3):213–222, 2021

[161] M. Vinueza-Morales, J. Córdova-Morán y J. Rodas-Silva. El uso del software alice como herramienta para el aprendizaje de programación: una revisión de literatura. En *Proceedings of the LACCEI international Multi-conference for Engineering, Education and Technology*, 2019

[162] M. Vinueza-Morales, J. Rodas-Silva, A. Chacón-Luna y H. Serrano Mantilla. Enseñanza de programación mediante mit app inventor: Una revisión de literatura. En *Proceedings of the LACCEI international Multiconference for Engineering, Education and Technology*, 2020

[163] J. Voogt, P. Fisser, J. Good, P. Mishra y A. Yadav. Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4):715–728, Dec 2015

[164] X. P. Voon, S. L. Wong, L.-H. Wong, M. N. M. Khambari y S. I. S. Syed-Abdullah. Developing computational thinking competencies through constructivist argumentation learning: A problem-solving perspective. *International Journal of Information and Education Technology*, 12(6): 529–539, 2022

[165] M. Wang y Y. Feng Wang. A study on computer teaching based on computational thinking. *International Journal of Emerging Technologies in Learning (iJET)*, 11:72, 12 2016

[166] P. S. Wang. *From Computing to Computational Thinking*. Chapman and Hall/CRC, 2017

[167] P. Wang, R. Bednarik y A. Moreno. During Automatic Program Animation, Explanations After Animations Have Greater Impact Than Before Animations. En *Koli*, Koli Calling '12, páginas 100–109, New York, NY, USA, 2012. ACM

[168] B. Ward, D. Marghitu, T. Bell y L. Lambert. Teaching computer science concepts in scratch and alice. *Journal of Computing Sciences in Colleges*, 26:173–180, 12 2010

[169] D. C. Webb. Troubleshooting assessment: an authentic problem solving activity for it education. En *WCLTA*, volumen 9, páginas 903–907, 2010

[170] D. Weintrop, A. K. Hansen, D. B. Harlow y D. Franklin. Starting from scratch: Outcomes of early computer science learning experiences and implications for what comes next. En *Proceedings of the 2018 ACM Conference on International Computing Education Research*, ICER '18, páginas 142–150, New York, NY, USA, 2018. ACM

[171] D. Weintrop y U. Wilensky. How block-based languages support novices a framework for categorizing block-based affordances. *Journal of Visual Languages and Sentient Systems*, 3:92–100, 01 2017

[172] L. Werner, S. Campe y J. Denner. Children learning computer science concepts via alice game-programming. En *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, página 427–432, New York, NY, USA, 2012. Association for Computing Machinery

[173] J. Wing. Computational thinking. *Communications of the ACM*, 49: 33–35, 03 2006

[174] J. Wing. Computational thinking and thinking about computing. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 366:3717–25, 11 2008

[175] L. S. Woei, I. H. Othman y C. K. Man. Learning programming using objects-first approach through folktales. *JUTE*, 75(3):47–53, 2015

[176] C. Wohlin, E. Mendes, K. Romero Felizardo y M. Kalinowski. Guidelines for the search strategy to update systematic literature reviews in softwa-

re engineering. *Information and Software Technology*, 127:106366, 2020

[177] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell y A. Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012

[178] B. Wu, Y. Hu, A. Ruis y M. Wang. Analysing computational thinking in collaborative programming: A quantitative ethnography approach. *Journal of Computer Assisted Learning*, 35(3):421–434, 2019

[179] B. Xie y H. Abelson. Skill progression in mit app inventor. páginas 213–217, 09 2016

[180] Z. Xu, A. Ritzhaupt, F. Tian y K. Umapathy. Block-based versus text-based programming environments on novice student learning outcomes: a meta-analysis study. *Computer Science Education*, páginas 1–28, 01 2019

[181] A. Yadav y M. Berges. Computer science pedagogical content knowledge: Characterizing teacher performance. *ACM Transactions on Computing Education*, 19:1–24, 05 2019

[182] A. Yadav, J. Good, J. Voogt y P. Fisser. *Computational Thinking as an Emerging Competence Domain*, páginas 1051–1067. Springer International Publishing, Cham, 2017

[183] O. Yasar. The essence of computational thinking. *Computing in Science Engineering*, 19(04):74–82, jul 2017

[184] I. Yoon, J. Kim y W. Lee. The analysis and application of an educational programming language (RUR-PLE) for a pre-introductory computer science course. *JCC*, 19(1):529–546, 2016

[185] E. Yukselturk y S. Altiok. An investigation of the effects of programming with Scratch on the preservice IT teachers' self-efficacy perceptions and attitudes towards computer programming. *BJET*, páginas n/a—-n/a, 2016

This document was typeset on 2022/6/22 using RC-BOOK α2.11 for LaTeX$_{2\varepsilon}$.
Should you want to use this document class, please send mail to
contact@tdg-seville.info.