Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

Estrategia didáctica de enseñanza y aprendizaje para programadores de software

Teaching and learning didactic strategy for software programmers

Ensino e aprendizagem de estratégia didática para programadores de software

Elvira Ivone González Jaimes
Universidad Autónoma del Estado de México, México
ivonegj@hotmail.com
https://orcid.org/0000-0002-5328-5586

Asdrúbal López Chau
Universidad Autónoma del Estado de México, México
alchau.uaemex@gmail.com
https://orcid.org/0000-0001-5254-0939

Valentín Trujillo Mora
Universidad Autónoma del Estado de México, México
vtrujillom@uaemex.mx
https://orcid.org/0000-0002-5936-4795

Rafael Rojas Hernández
Universidad Autónoma del Estado de México, México
rrojashe@uaemex.mx
https://orcid.org/0000-0001-6649-067X

Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

Resumen

Introducción. Las estrategias didácticas utilizadas para la enseñanza y el aprendizaje de la programación de *software* son complejas porque se debe adquirir, codificar y recuperar información con base en el pensamiento lógico-matemático para diseñar instrucciones que serán ejecutadas por un ordenador con el fin de resolver problemas de diversa naturaleza. Esto justifica por qué estos contenidos tienen alta demanda en la educación superior de la actualidad, a pesar de que los índices de titulación sean, por el contrario, muy bajos. **Objetivo.** Proponer y evaluar estrategias didácticas generales y específicas para la enseñanza y aprendizaje de programación en *software* con el fin de crear una alternativa para disminuir los registros de bajo rendimiento y la deserción en las carreras que incluyan temas de la mencionada área. **Método.** Diseño cuasiexperiemtal, análisis cuantitativo no paramétrico, longitudinal (cuatro evaluaciones), con una muestra de 78 estudiantes de la carrera de ingeniería en Computación, con distribución al azar en dos grupos experimentales y dos grupos de control en escenarios de aulas inteligentes. **Material.** Programas de *software* con lenguajes C y C++ en plataforma educativa, así como uso de Autogradr y del *Inventario de estrategias metacognitivas*. **Resultados.** Se empleó una prueba de Kruskal-Wallis debido a que se tuvo una muestra sin distribución normal y con escala por jerarquías. Se observó la aceptación de la hipótesis nula en ambas pruebas: 1) H = 0.04, en rendimiento académico superior en los grupos experimentales, la interdependencia en conocimientos es regular alta 0.657, con diferencia de 1.7 en rendimiento promedio, y 2) H = 0.24, en estrategias metacognitivas superior en los grupos experimentales, la interdependencia en estrategias metacognitivas es baja 0.342. Se resalta el área de planificación como fase elemental para la solución de problemas. **Conclusiones.** Se demostró que la estrategia didáctica expuesta en tres bloques específicos de enseñanza y aprendizaje elevó el rendimiento académico y la metacognición de los estudiantes.

Palabras claves: estrategias de aprendizaje, estrategia de enseñanza, programas de computación.

Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

Abstract

Introduction. The didactic strategies used for teaching and learning software programming are complex because information must be acquired, coded and retrieved based on logical-mathematical thinking to design instructions that will be executed by a computer in order to solve problems of diverse nature. This justifies why these contents are in high demand in today's higher education, despite the fact that the degree indices are, on the contrary, very low. **Objective**. Propose and evaluate general and specific didactic strategies for teaching and learning software programming in order to create an alternative to reduce low-performance records and dropout in careers that include subjects from the above area. **Method.** Design quasiexperiemtal, quantitative analysis not parametric, longitudinal (four evaluations), with a sample of 78 students of the career of engineering in Computation, with random distribution in two experimental groups and two groups of control in scenarios of intelligent classrooms. **Material.** Software programs with C and C++ languages in educational platform, as well as the use of Autogradr and the Inventory of metacognitive strategies. **Results**. A Kruskal-Wallis test was used because we had a sample without normal distribution and with scale by hierarchies. The acceptance of the null hypothesis was observed in both tests: 1) H = 0.04, in superior academic performance in the experimental groups, the interdependence in knowledge is regular high 0.657, with difference of 1.7 in average performance, and 2) H = 0.24, in superior metacognitive strategies in the experimental groups, the interdependence in metacognitive strategies is low 0.342. The planning area is highlighted as an elementary phase for the solution of problems. **Conclusions**. It was demonstrated that the didactic strategy exposed in three specific blocks of teaching and learning elevated the academic performance and the metacognition of the students.

Keywords: learning strategies, teaching strategy, computer programs.

Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

Resumo

Introdução As estratégias didáticas utilizadas para o ensino e aprendizagem de programas de software são complexas, pois devem adquirir, codificar e recuperar informações baseadas em raciocínio lógico-matemático para projetar instruções que serão executadas por um computador para solucionar problemas de palavras. natureza diversa. Isso justifica por que esses conteúdos estão em alta demanda no ensino superior de hoje, embora os índices de grau sejam, ao contrário, muito baixos. Objetivo Propor e avaliar estratégias didáticas gerais e específicas para o ensino e aprendizagem de programação de software, a fim de criar uma alternativa para reduzir os registros de baixo desempenho e abandono em carreiras que incluam tópicos da área acima mencionada. Método Projeto Cuasiexperiemtal, análise quantitativa não paramétrica, longitudinal (quatro avaliações), com uma amostra de 78 alunos da carreira de engenharia em Computação, com distribuição aleatória em dois grupos experimentais e dois grupos controle em ambientes de sala de aula inteligente. Material Programas de software com linguagens C e C ++ na plataforma educacional, bem como o uso do Autogradr e do Inventário de Estratégias Metacognitivas. Resultados Um teste de Kruskal-Wallis foi usado porque havia uma amostra sem distribuição normal e escala por hierarquias. A aceitação da hipótese nula foi observada em ambos os testes: 1) H = 0,04, em maior desempenho acadêmico nos grupos experimentais, a interdependência no conhecimento é alta regular 0,657, com diferença de 1,7 no desempenho médio, e 2) H = 0,24 , nas estratégias metacognitivas mais elevadas nos grupos experimentais, a interdependência nas estratégias metacognitivas é baixa 0.342. A área de planejamento é destacada como a fase elementar para resolver problemas. Conclusões Foi demonstrado que a estratégia didática exposta em três blocos específicos de ensino e aprendizagem elevou o desempenho acadêmico e a metacognição dos alunos.

Palavras-chave: estratégias de aprendizagem, estratégia de ensino, programas de computador.

Fecha Recepción: Noviembre 2017 **Fecha Aceptación:** Mayo 2018

Introducción

Las estrategias didácticas utilizadas para adquirir, codificar y recuperar información en programadores de *software* son complejas, ya que exigen no solo un pensamiento lógico-matemático para formular algoritmos y diseños matemáticos, sino también la capacidad de retención y manipulación del lenguaje informático. Por tal motivo, en este estudio se han seleccionado, expuesto y probado estrategias didácticas para procurar facilitar la adquisición de conocimientos de programadores de *software*. Para esto, se ha tomado como sustento el estudio previo de González y López (en prensa) donde se observaron las necesidades específicas que se requieren para programar en lenguaje C, específicamente en un escenario de aula inteligente (*smart classrrom*).

Sobre este aspecto, vale destacar que las actividades que comúnmente realiza el programador de *software* a nivel profesional son construir instrucciones que serán ejecutadas por un ordenador para resolver diversos problemas de la vida diaria, de ahí que se justifique por qué este conocimiento ha empezado a tener una alta demanda en la educación superior (Kori, Pedaste, Altin, Tõnisson, y Palts, 2016).

El problema, sin embargo, se centra en los altos niveles de deserción y reprobación durante los primeros años de la carrera, como lo demuestran las cifras de distintos estudios. En efecto, según Kori *et al.* (2015), en Estonia 32.2 % de los alumnos de primer año abandonaron durante 2015 las carreras que se relacionaban con la informática y las tecnologías de la información. Igualmente, en una indagación realizada en quince universidades españolas vinculadas con en la mencionada área se determinó que durante el primer año del período 2015-2016 los niveles de fracaso escolar se encontraban en 22.5 % (Universia España, 2017).

En Latinoamérica los resultados son similares: Lázaro, Callejas, Griol y Durán (2017), por ejemplo, llevaron a cabo un estudio en Colombia en el cual se demostró que entre 45 % y 52 % de los estudiantes que ingresaron a un programa de ingeniería en Informática lo habían abandonado. Asimismo, y en el caso específico de la Universidad de Quindío, Castro, Hernández y Vanegas (2013) reportaron que en esa casa de estudio los alumnos solo lograban acreditar 27 % de las materias básicas, razón por la cual los índices de deserción se encontraban en 42 %. Siguiendo con el caso de Colombia, pero específicamente en la Universidad Simón

Bolívar de Barranquilla, Azoumana (2013) empleó la minería de datos para señalar que entre 2007 y 2012 el porcentaje de deserción se hallaba en 65 %.

En Argentina, por otra parte, Ferrero y Oloriz (2015) realizó un estudio longitudinal en la Universidad Nacional de Luján durante el periodo 2000-2010, en el cual encontró una alta correlación entre el rendimiento académico de los estudiantes en matemática y el abandono de 71 % durante el primer año de las carreras de Ingeniería. Igualmente, en Perú Sánchez, Gómez y Gaviria (2016) desarrollaron un trabajo en la Universidad de la Amazonia, en el cual hallaron que en 2012 los niveles de deserción se ubicaban en 9 %, mientras que en 2012 esa cifra ascendió a 20 %. En México, por último, la Universidad Autónoma del Estado de México (UAEM, 2017) reportó que 74.2 % de los alumnos de primer año de esa institución reprobaban los exámenes finales (índice que disminuía a 30.4 % en el quinto año), mientras que la eficiencia terminal global se ubicaba en 51.4 %, el índice de titulación se hallaba en 43.2 % y el índice de abandono escolar estaba en 12.35 %.

Requerimientos personales para aprender a programar

El aprendizaje de nuevos lenguajes de programación C exige el desarrollo de la memoria a corto y largo plazo para recordar los signos y sus respectivos significados, lo cual se puede conseguir de manera similar a como se adquiere cualquier lenguaje, aunque añadiendo la práctica del razonamiento lógico-matemático. Para esto, por supuesto, se puede contar con el acompañamiento cercano de un tutor, quien puede evitar que el alumno continúe editando errores mientras se encuentra detenido en el proceso de abstracción inductiva y deductiva, lo cual interrumpe el acomodo de la información y la autorregulación del aprendizaje (González y López, en prensa; González, López, Rojas y Trujillo, en prensa).

Qué se va a aprender. Programación de software en lenguaje C y C++

El presente estudio se enfoca en el desarrollo del aprendizaje del lenguaje de programación en C y su aplicación en C++, los cuales tienen diversas ventajas, como se señalan a continuación: 1) los lenguajes C y su evolución C++ son poderosos y flexibles, con órdenes, operaciones y funciones de biblioteca que se pueden utilizar para escribir la mayoría de los programas que corren en diversos diseños de computadoras, 2) son usados por

profesionales para desarrollar *softwares* en la mayoría de los modernos sistemas de computadora, 3) son empleados para desarrollar sistemas operativos, compiladores, sistemas de tiempo real y aplicaciones de comunicaciones, 4) son portables y se pueden trasladar de un tipo de computadora a otra, y 5) tienen una alta velocidad de ejecución y son casi universales porque existen bibliotecas del lenguaje C y C++ que soportan gran variedad de aplicaciones como bases de datos gráficos, edición de texto, comunicaciones, etc. (Joyanes y Zahonero, 2005).

En dónde se va a aprender (escenario). Aula inteligente (smart classrrom)

El aprendizaje de programación de *software* en un aula inteligente o *smart classroom* tiene sus ventajas; por ejemplo, 1) aumento de la atención en los estudiantes y creación de un entorno interactivo y colaborativo (Lozano, 2004; Antona *et al.*, 2011), 2) acceso reflexivo ante la solución de problemas e implementación de las soluciones en un lenguaje de programación en C (Cairó, 2006), 3) interactividad en la adquisición de conocimientos, proceso de información y transformación del aprendizaje en un producto (Cook, Augusto y Jakkula, 2009), y 4) en el momento en que se puede palpar y valorar el conocimiento, permite reflexionar sobre lo alcanzado, lo que es útil para proponer autodiálogos estructurados y para evaluar el porqué, el cómo y el para qué, lo que estimula la metacognición del estudiante (Bravo, Hervás, Sánchez y Crespo, 2004). Además, existen principios universales del aula inteligente que se pueden usar por su configuración pedagógica generalizables (Maheshwari, 2016).

Cómo se va a aprender. Proceso de aprendizaje de conocimientos complejos

Después de considerar qué y en dónde se va a aprender, es importante reflexionar sobre cómo se aprende, pues de esta manera se puede no solo entender la manera en que el sujeto conoce al objeto a través de un aula inteligente (Sánchez, 2000), sino también conocer cuáles estrategias de aprendizaje deben ser utilizadas (González y López, en prensa). De esta manera, se promueve a través de un nuevo paradigma educativo la metacognición de elementos complejos en un aula inteligente que intenta facilitar y acelerar la adquisición, codificación y reproducción del conocimiento (Lozano, 2004). Para ello, sin embargo, se debe disponer de

un conjunto de habilidades relacionadas "básicamente con los procesos ligados a la adquisición, organización, retención y uso del conocimiento" (Gutiérrez, 2005, p. 6).

La adquisición se vincula con el mapa mental que posee el sujeto y la interrelación con el nuevo conocimiento, proceso en el cual se asimila, transforma y reorganiza el pensamiento hasta conseguir la retención de conceptos nuevos. Luego, cuando el individuo lo regresa al mundo, surge la adaptación y el acomodamiento: "dos poderosos motores que hacen que el ser humano mantenga ese desarrollo continuo de sus estructuras cognitivas" (Pinto y Martínez, 1994, p. 25).

La metacognición, entonces, es una piedra angular porque se sustenta en tres aspectos esneciales:1) la toma de conciencia, es decir, el proceso de conceptualización y manipulación del objeto (uso de códigos y procesos lógicos del flujo de los diagramas), 2) el proceso de abstracción, el cual se presenta en cualquier etapa del desarrollo y permite el acceso a un nuevo conocimiento (poder realizar operaciones simples o complejas), y 3) la autorregulación, donde el sujeto compensa o nivela las perturbaciones cognitivas, contratiempos o molestias que debe enfrentar ante el objeto en el aprendizaje (facilitado por los dispositivos electrónicos) (González y López, en prensa).

Selección de estrategias didácticas como intervención para facilitar el conocimiento

El concepto de estrategias didácticas "involucra la selección de actividades y prácticas pedagógicas en diferentes momentos formativos, métodos y recursos en los procesos de enseñanza aprendizaje" (Velazco y Mosquera, 2015, pp. 1-2). La clasificación de estas son diversas, pues se deben tomar en cuenta distintas variables; por ejemplo, el objetivo de estudio, es decir, enseñanza y aprendizaje de programas de *software* en lenguajes C y C++ (Joyanes y Zahonero, 2005). Asimismo, el escenario de estudio, esto es, 1) aulas inteligentes con conexión permanente a internet de banda ancha, 2) plataformas con *softwares* educativos diseñadas por expertos en programación de lenguaje C y C++ (Santana-Mancilla, Magaña, Rojas, Nieblas y Salazar, 2013; Sharma, 2016), 3) laboratorio inteligente o *smart lab* para la enseñanza de la programación (Alammary, Carbone y Sheard, 2012), y 4) herramientas didácticas como Autogradr (Soni y Dalal, 2018).

Planteado lo anterior, en este trabajo se emplearon estrategias de enseñanza y aprendizaje *generales* (para ayudar a adquirir todo tipo de conocimiento) y *específicas* (según el factor contextual, es decir, aula inteligente, plataforma educativa, laboratorio inteligente y herramienta didáctica Autogradr, las cuales resultan útiles en la tarea de aprender el lenguaje de programación C y C++) (Legorreta, 2015). A continuación, se explica cada una de estas:

Estrategias generales

En cada uno de los bloques el profesor planteó a) los objetivos y metas, b) las fichas de conceptos con significado, leyes, reglas y principios, c) las lecturas con lenguaje de programación y d) los grupos de apoyo.

Estrategias específicas

El primer bloque se focalizó en el escenario *aulas inteligentes*, el cual tenía su principio pedagógico de multiplicidad, que posibilita al mismo tiempo el uso de varios tipos de recursos y estímulos (Maheshwari, 2016). Para evitar la sobreestimulación (disparador de la distracción) se utilizaron las estrategias de enseñanza y aprendizaje de analogía, las cuales permiten fijar la atención y anclarla al conocimiento previo. Para fijar el conocimiento nuevo se utilizó la estrategia de codificación (nemotecnización) como proceso subsiguiente a la adquisición; esto implica un procesamiento más profundo y complejo porque se integra a la información previa en estructuras de significado más amplias, lo cual sirve para la elaboración de mapa conceptuales y diagramas de flujo (Díaz y Hernández, 2002; Flores y Juárez, 2017).

En el segundo bloque se utilizó la estrategia de exploración, mediante la cual se les ofrecieron a los alumnos páginas de lecturas de lenguajes en C y C++ (según el nivel de avance, en laboratorios inteligentes) y se les motivó para que compartieran nuevos textos que fueran útiles para reforzar la secuencia y fluidez (Cairó, 2006). Igualmente, se emplearon recursos externos (lecturas sugeridas en los laboratorios inteligentes) y recursos internos (formas de investigar). A partir de esto, el estudiante pudo seguir y construir secuencias didácticas (entretejiendo actividades en laboratorio) que favorecieron sus aprendizajes significativos, pues se consiguió enlazar lo aprendido en la clase con lo investigado en los sistemas electrónicos (Díaz-Barriga, 2013).

En el tercer bloque (donde se usó la llamada *estrategia de fragmentación*) se le ofreció al alumno un todo para que lo separara en niveles de importancia y luego en pedazos, de esta manera se procuró la integración en secuencias lógicas y algorítmicas. En otras palabras, primero se brindan ejercicios de la estrategia de fragmentación *idiosincrásica*, en donde se inicia del todo para desmenuzarlo en partes y para analizar los fragmentos, y luego se utiliza la estrategia de fragmentación *epigrafiada* para reconstruir ese todo según sus características. Así, se evitan los vacíos en el aprendizaje (Joyanes y Zahonero, 2005). Simultáneamente, se empleó la herramienta educativa para programadores Autogradr, la cual posee organización, metodología y recursos esquemáticos como modelo para elaborar algoritmos y ejecutar el diseño, lo cual favorece las bases para la resolución de un problema en programación. Esta herramienta educativa tiene un impacto directo sobre el diseño de instrucciones de programación porque mejora el nivel de funcionamiento cognitivo.

Por otra parte, y de acuerdo con los antecedentes del presente objeto de estudio (aprendizaje del lenguaje de programación en C y C++ con uso de la tecnología), el marco teórico usado fue el reconstructivismo (fundado en el constructivismo), el cual permite sumar diferentes posturas pedagógicas, psicológicas y sociales que lo enriquecen y actualizan para el aprendizaje significativo (Díaz y Hernández, 2002). Este, además, se encuentra en sintonía con la metacognición, elemento que juega un papel principal en el aprendizaje significativo porque permite la selección y regulación inteligente de estrategias y técnicas de aprendizaje, lo cual impacta de forma positiva no solo en la motivación y la concentración de las tareas propuestas, sino también en la organización del tiempo de estudio, aspectos que se habían detectado como principales causantes del bajo rendimiento de los alumnos.

Por último, se debe resaltar el rol del docente o tutor (experto y especialista) como conocedor de los procesos de desarrollo intelectual y de las capacidades cognitivas en las diversas etapas de aprendizaje del estudiante.

Antecedentes de los materiales para la evaluación

En primer lugar, se debe mencionar el *Inventario de estrategias metacognitivas* elaborado por O'Neil y Abedi (1996), el cual fue traducido y adaptado al español por Martínez (2004) para analizar las estrategias de aprendizaje que emplean los estudiantes universitarios en el desarrollo de las tareas. Este focaliza las actividades metacognitivas en cuatro dimensiones (conciencia, estrategias cognitivas, planificación y control) y se apoya en la técnica del análisis factorial, la cual permite explicar el fenómeno complejo de la metacognición de una manera precisa y sensible, a través de la validez de constructo de los ítems por su correlación y homogeneidad comprobada (Vallejos, Jaimes, Aguilar y Merino, 2012).

- Ficha técnica: Adultos, nivel académico universitario, administración individual y grupal, con tiempo promedio de aplicación de 20 minutos.
- Características de la prueba: Veinte ítems evaluados con una escala tipo Likert, con cinco opciones de respuesta (1 = nunca, 2 = pocas veces, 3 = regular, 4 = muchas veces, 5 = siempre) y evaluación por jerarquía (bajo, mediano, alto y muy alto) correspondientes a los percentiles 20, 40, 60 y 80.
- Índices de confiabilidad α .90 y varianza de 40.81, obtenidos por análisis factorial exploratorio de máxima verosimilitud y rotación oblimin directo (Vallejos *et al.*, 2012)

Autogradr, por otra parte, es una herramienta educativa diseñada para apoyar la enseñanza y el aprendizaje en las ciencias de la computación (en específico, para programación) y para crear asignaciones personalizadas de programación. Además, permite evaluar de forma automática los códigos fuente de los alumnos y brindar retroalimentación instantánea a los estudiantes. De esta forma, se acorta el ciclo de interacción entre los instructores y los estudiantes. Por otro lado, si el código fuente no genera las respuestas correctas, Autogradr indica exactamente en dónde se encuentra el error, lo que ayuda a depurar los programas y a realizar varios intentos hasta solucionar el problema (Soni y Dalal, 2018).

Autogradr soporta la creación de dos tipos de actividades que pueden ser asignadas a los alumnos. La primera son los laboratorios, en los que el alumno introduce el código fuente en el editor de Autogradr. La segunda son los proyectos, en los que el alumno puede subir uno

o más archivos con el código fuente. Los laboratorios están pensados para la práctica de temas aislados, mientras que los proyectos para temas más complejos o que requieren la aplicación de diversos conceptos. Esto genera índices de rendimiento en cuanto al número de aciertos obtenidos en los laboratorios o proyectos asignados (Soni y Dalal, 2018).

Por último, el tercer tipo es la plataforma educativa con la materia Programación de Lenguaje C y C++, lo que ofrece índices y aciertos de las actividades asignadas en un portafolio (Roa, 2015).

Metodología

El presente fue un estudio cuasiexperimental, de campo, comparativo, con análisis cuantitativo no paramétrico y estadística descriptiva e inferencial, realizados en cuatro grupos (dos experimentales y dos de control). La población y muestra fue la misma porque se trabajó con todos los 78 estudiantes inscritos en el primer y tercer semestre de ambos turnos (matutino y vespertino) de la carrera de ingeniería en Computación. La asignación de los grupos fue elegida al azar porque ya que se había seleccionado por sorteo uno los turnos.

Los dos grupos experimentales poseían cuatro registros: dos longitudinales (semanales: índices de aciertos en la herramienta educativa Autogradr, y mensuales: índice de actividades realizadas del portafolio de evidencia de la plataforma educativa), y dos transversales (semestral: aplicación del *Inventario de estrategias metacognitivas*, y semestral: índices de rendimiento académico). En cambio, los dos grupos de control tenían dos registros transversales: uno semestral (aplicación del *Inventario de estrategias metacognitivas*) y uno semestral (índices de rendimiento académico). En cuatro grupos se usaron las mismas condiciones físicas (aula inteligente).

En el análisis cualitativo descriptivo se determinaron 1) las características de la muestra, 2) la frecuencia en porcentaje a $n \geq 80$ % en actividades de herramienta educativa Autogradr y 3) el portafolio de evidencia, en tres estrategias de aprendizaje en porcentaje a $n \geq 80$ %.

Igualmente, se realizó el análisis cuantitativo inferencial, no paramétrico, prueba de Kruskal-Wallis, para conocer la diferencia entre los cuatro grupos independientes, sin distribución normal. En tal sentido, tiene escala por jerarquías del *Inventario de estrategias*

metacognitivas y escala continua en los índices de rendimiento académico (Dawson-Saunders y Trapp, 2002).

Hipótesis

H_1: Los cuatro grupos son idénticos en cuanto a los índices de rendimiento académico.

H_0: Los cuatro grupos no son idénticos en cuanto a los índices de rendimiento académico.

H_2: Los cuatro grupos son idénticos en cuanto a los índices del *Inventario de estrategias metacognitivas*.

H_0 Los cuatro grupos no son idénticos en cuanto a los índices del *Inventario de estrategias metacognitivas*.

Regla de decisión:

Si $X^2_c \geq X^2_{tab}$ se rechaza la H_1

X_{2tab} con K -1 grados de libertad (4-1) = 3

Procedimiento

- Fase 1: Consentimiento informado. Firma de la carta por estudiantes.
- Fase 2: Aplicar la intervención en dos grupos experimentales (primer y tercer semestre). Dos horas teóricas con estrategias de aprendizaje, dos horas de prácticas en aula inteligente y dos horas utilizando la herramienta educativa Autogradr (tabla 1).

Tabla 1. Estrategias de aprendizaje

Adquirir	Asimilar y transformar Proceso interno	Reproducir	Portafolio de evidencias
Estrategias de atención. Estrategias de exploración.	Estrategias de analogías y codificación, integrando palabras o imágenes. Estrategia de elaboración, propio lenguaje y acciones.	Estrategias de búsqueda de codificaciones (nemotecnia, mapas, matrices, secuencias, etc.)	Evaluación de mapas conceptuales y diagramas de flujo. Evaluación de lecturas en lenguaje de programación.
Estrategias de fragmentación: analizar las partes de un todo organizado; primero, identificar y resaltar lo relevante (idiosincrásico). Segundo, asignar sentido lógico matemático (epigrafiado).	Estrategia de organización: Separa por características o significados (semántica) y luego lo junta en conjuntos con reglas (síntesis), realiza esquemas, secuencias lógicas y secuencias temporales	Generación: Tratar de formular una respuesta por 1) ensayo y error, o libre asociación, 2) ordenación de conceptos, y 3) escribir la respuesta hasta que se ajuste a la imagen reconocida.	Evaluación de la secuencias lógica algorítmica para la solución de problemas en programación.
Estrategias de repetición y corrección: Usar programa Autogradr.	Darse cuenta de aciertos y errores para corrección.	Realización de ejercicios identificando el acierto o error hasta lograr respuestas correctas a la solución de problemas.	Evaluación de actividades de éxito del programa Autogradr

Fuente: Elaboración propia

En grupo de control, dos horas teóricas y cuatro horas de prácticas en aula inteligente.

- Fase 3: Evaluar intervención. 1) Semanal: evaluación de actividades de herramienta educativa Autogradr, con un total de 80 % de asertividad y 2) mensual en portafolio de evidencia de estrategias de aprendizaje; en esta se incluyen a) estrategia de atención, evaluación de mapas conceptuales y diagramas de flujo, b) estrategia de exploración,

evaluación de lecturas en lenguaje de programación y c) estrategia de fragmentación, evaluación de la secuencia lógico-algorítmica para la solución de problemas en programación. En cada una de las estrategias se requiere un total de 80 % de asertividad.

- Fase 4: Aplicar y evaluar. 1) Semanal: actividades de herramienta educativa Autogradr y 2) mensual: en portafolio de evidencia de las tres estrategias de aprendizaje.
- Fase 5: Aplicar y evaluar. Semestral: *Inventario de estrategias metacognitivas*.
- Fase 6: Recolectar. Semestral: índices de rendimiento académico en control escolar.
- Fase 7: Análisis de los datos. Análisis cualitativo y descriptivo de 1) características de la muestra, 2) frecuencia de actividades de herramienta educativa Autogradr, 3) portafolio de evidencia y 4) índices de rendimiento académico. Análisis cuantitativo inferencial, prueba de Kruskal-Wallis en el *Inventario de estrategias metacognitivas* y rendimiento académico (Dawson-Saunders y Trapp, 2002).

Resultados

El análisis descriptivo se centró en:

1. Características de la muestra: Edad promedio = 21.4 años; género = 85 % masculino y 15 % femenino; turnos = 54 % aprendizaje programa C y 46 % aprendizaje programa C++.
2. Evaluación semanal de actividades realizadas en la herramienta educativa Autogradr. Se solicitó 80 % de asertividad para acreditar (tabla 2).

Tabla 2. Porcentaje promedio de rendimiento en la herramienta educativa Autogradr

	Grupo experimental 1: aprendizaje programa C n = 42.54 %		Grupo experimental 2: aprendizaje programa C++ n = 36.46 %	
	Laboratorio	Proyecto	Laboratorio	Proyecto
1 a 4 semana	81 %	71 %	83 %	65 %
5 a 8 semana	84 %	79 %	84 %	72 %
9 a 12 semana	85 %	82 %	84 %	79 %
13 a 16 semana	86 %	85 %	82 %	80 %

Fuente: Elaboración propia

3. Evaluaciones mensuales del portafolio de evidencia con estrategias de aprendizaje. Se incluyó a) estrategia de atención (evaluación de mapas conceptuales y diagramas de flujo), b) estrategia de exploración (evaluación de lecturas en lenguaje de programación) y c) estrategia de fragmentación (evaluación de la secuencias lógica algorítmica para la solución de problemas en programación). En cada una de las estrategias se requiere un total de 80 % de asertividad para acreditar (tabla 3).

Tabla 3. Porcentaje promedio de evidencias en portafolio

	Grupo experimental 1: programa C n = 32.48 %			Grupo experimental 2: programa C++ n = 35.52 %		
	Estrategia de atención	Estrategia de exploración	Estrategia de fragmentación	Estrategia de atención	Estrategia de exploración	Estrategia de fragmentación
1.er mes	84 %	70 %	54 %	82 %	68 %	73 %
2.° mes	76 %	74 %	61 %	78 %	70 %	73 %
3.er mes	81 %	75 %	60 %	83 %	72 %	74 %
4.° mes	86 %	81 %	74 %	85 %	81 %	81 %

Fuente: Elaboración propia

4. Pruebas de hipótesis para conocer si los cuatro grupos son iguales o diferentes en a) el rendimiento académico y b) en el *Inventario de estrategias metacognitivas*. La decisión está en función de los resultados calculados al cuadrado: si es mayor o igual al resultado que está en las tablas, entonces se rechaza la hipótesis verdadera, lo que significa que los cuatro grupos son diferentes. Obteniendo en el programa SPSS, versión 17. Entre los cuatro grupos, medir el rendimiento académico.

   Valor del estadístico de Kruskal-Wallis (H = 0.04) en rendimiento académico superior en los grupos experimentales, por lo que se rechaza la hipótesis verdadera, aunque la interdependencia en conocimientos es regular alta 0.657 (tabla 4)

Tabla 4. Estadísticas de contraste

	Rendimiento académico	Interdependencia
chi cuadrada	8.86	0.745
grados de libertad	3	3
sing. asintót	0.04	0.657

Fuente: Elaboración propia

Valor del estadístico de Kruskal-Wallis (H = 0.24) en estrategias metacognitivas superior en los grupos experimentales, por lo que se rechaza la hipótesis verdadera y su interdependencia en estrategias metacognitivas, que también es baja 0.342 (tabla 5).

Tabla 5. Estadísticas de contraste

	Estrategias metacognitivas	Interdependencia
chi cuadrada	60.65	0.423
grados de libertad	3	3
sing. asintót	0.24	0.342

Fuente: Elaboración propia

Discusión

En el presente estudio se probó una metodología didáctica que incluyó estrategias de enseñanza y aprendizaje específicas para programadores de sistemas de *software* como una alternativa de solución a los bajos índices de aprobación, motivo por el cual ha aumentado el abandono de la carrera ingeniería en Computación en una universidad pública de México.

Ahora bien, en cuanto a las limitaciones de este trabajo, se puede señalar que tanto la población como la muestra seleccionadas fueron las mismas, las cuales correspondieron a la matrícula del primer y tercer semestre de la mencionada carrera.

Por otra parte, en relación con las fortalezas, se pueden señalar las siguientes: la estrategia didáctica se basó en un estudio previo que permitió detectar las necesidades específicas y las estrategias de aprendizaje que se debían utilizar para adquirir, codificar y reproducir el lenguaje de programación C en un escenario de aula inteligente (González y López, en prensa).

Asimismo, y por ser poderosos y flexibles, se utilizaron los lenguajes de programación C y C++, los cuales son empleados frecuentemente por desarrolladores de sistemas operativos debido a sus características específicas de uso y portabilidad (Joyanes y Zahonero, 2005). Estos fueron ejecutados en aulas inteligentes, las cuales han sido diseñadas sobre principios pedagógicos que facilitan y aceleran el aprendizaje (Bravo, Hervás y Chavira, 2005).

Las primeras estrategias de aprendizaje se basaron en la necesidad de fijar la atención en conceptos nuevos para luego vincularlos con los registros previos. En tal sentido, en clase se les mostraron a los alumnos estrategias de analogías y codificación (personalizando el lenguaje de cómputo) que les ayudaría a integrar información y a elaborar mapas conceptuales y diagramas de flujo (Díaz y Hernández, 2002). Con estas estrategias se logró el objetivo fijado en las actividades: grupo experimental 1 con 86 % y grupo experimental 2 con 85 %.

La segunda estrategia fue la de exploración, mediante la cual se les brindaron a los alumnos páginas para realizar lecturas del lenguaje de programación C y C++ (según el nivel de avance), con lo que se reforzó la secuencia y fluidez (Cairó, 2006). Con esta estrategia se logró el objetivo requerido cumplimiento de las actividades: grupo experimental 1 con 81 % y grupo experimental 2 con 81 %.

La tercera estrategia fue la de fragmentación, la cual consistió en separar, clasificar e integrar secuencias lógico-algorítmicas para procurar hallar la solución a diversos problemas de programación. De esta manera se pudieron identificar los procesos completos desde la entrada hasta la salida (Joyanes y Zahonero, 2005). Con esta, sin embargo, no se logró el objetivo requerido cumplimiento de las actividades: grupo experimental 1 con 74 % y grupo experimental 2 con 78 %.

La cuarta estrategia (repetición y corrección) se basó en el uso del programa Autogradr, herramienta educativa diseñada para apoyar la enseñanza y el aprendizaje de las ciencias de la computación, en específico para programación. Esta resulta muy útil para evaluar y retroalimentar de forma automática los códigos fuente de los estudiantes, pues Autogradr indica exactamente en dónde se encuentra el error para que el alumno lo modifique. De esta manera, se ayuda a depurar los programas para solucionar el problema.

Con esta estrategia se logró el objetivo requerido (cumplimiento de las actividades) solo en el grupo experimental 1 (laboratorio 86 % y proyecto 85 %) y en el grupo experimental 2 (laboratorio 82 % y proyecto 80 %).

Otra de las herramientas propuestas continuamente fue la reflexión, es decir, la metacognición ante los procesos de aprendizaje y logros. Esto fue determinante para estimular el autoapoyo y la autorregulación, pues los alumnos requerían retroalimentación emocional, variable indispensable para que continuar con la motivación, ya que el aprendizaje en ciencias exactas se constituye sobre la base de altos niveles de dificultad (Barrón, Zatarain y Hernández, 2014; Flores y Juárez, 2017).

Por otra parte, para medir el nivel en el desarrollo de la metacognición, se utilizó el *Inventario de estrategias metacognitivas*, con el cual se evaluaron las siguientes dimensiones: conciencia, estrategias cognitivas, planificación y control (Vallejos *et al.*, 2012).

En cuanto al valor del estadístico de Kruskal-Wallis, H = 0.04, se obtuvo rendimiento académico superior en los grupos experimentales, por lo que se rechaza la hipótesis verdadera, aunque la interdependencia en conocimientos es regular-alta 0.657, y la diferencia en rendimiento es 1.7 en promedio.

Por último, en relación con el valor del estadístico de Kruskal-Wallis, H = 0.24, se consiguió en estrategias metacognitivas números superiores en los grupos experimentales, por lo que se rechaza la hipótesis verdadera; además, su interdependencia en estrategias metacognitivas es baja 0.342, y se resalta el área de planificación, fase elemental para la solución de problemas.

Conclusiones

Para el trabajo pedagógico con leguajes de programación C y C++ se requiere la detección de estrategias de aprendizaje y necesidades utilizadas, pues de esa manera se puede proponer una intervención más ajustada a la realidad. En este sentido, la estrategia didáctica elaborada y probada fue la siguiente: en el primer bloque se incluyen estrategias de aprendizaje por analogía y codificación para atender y fijar conocimientos. En el segundo bloque se emplean estrategias de exploración de lecturas de códigos de *software*, construcción de laboratorios y proyectos. En el tercer bloque se toma en cuenta la estrategia de fragmentación

para integrar entradas y salidas en solución de problemas. Todo lo anterior sumado a la repetición y detección de errores suministrados por la herramienta didáctica Autogradr.

La conclusión general, por tanto, es que el uso de la secuencia didáctica de estrategias de aprendizaje, así como la herramienta didáctica Autogradr expuesta en este estudio facilitan la adquisición e implementación del lenguaje universal C y C++ para estudiantes programadores de sistemas de *software*. Esto significa que la instrucción académica en esta rama de estudio es indispensable, porque los *softwares* son elementos ubicuos que proveen flexibilidad, inteligencia y seguridad a todos los sistemas complejos y equipos de uso diario, desde el encendido automático de las luces de una casa hasta el complejo tablero de una nave espacial.

Por ende, se debe recordar que la enseñanza adecuada de los lenguajes de programación va a disminuir los niveles de reprobación y abandono de los futuros ingenieros en los primeros semestres de la carrera, es decir, cuando se aprenden las bases de los lenguajes universales para computación.

Referencias

Alammary, A., Carbone, A. and Sheard, J. (2012). Implementation of a Smart Lab for Teachers of Novice Programmers. ACE '12 *Proceedings of the Fourteenth Australasian Computing Education*, Conference 123, 121–130. Retrieved from dl.acm.org/citation.cfm?id=2483731.

Antona, M., Leonidis, A., Margetis, G., Korozi, M., Ntoa, S. and Stephanidis, C. (2011). A Student-Centric Intelligent Classroom. *International Joint Conference on Ambient Intelligence*, 248–252. Doi: 10.1007/978-3-642-25167-2_33.

Azoumana, K. (2013). Análisis de la deserción estudiantil en la Universidad Simón Bolívar, Facultad Ingeniería de Sistemas, con técnicas de minería de datos. *Pensamiento Americano*, 41-51.

Barrón, M., Zatarain, R. y Hernández, Y. (2014). Tutor inteligente con reconocimiento y manejo de emociones para Matemáticas. *Revista Electrónica de Investigación Educativa*, *16*(3), 88-102. Recuperado de https://redie.uabc.mx/redie/article/view/954/966.

Bravo, J., Hervás, R. and Chavira, G. (2005). Ubiquitous Computing in the Classroom: An Approach through Identification Process. *Journal of Universal Computer Science*, *11*(9), 1494-1504. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.5542&rep=rep1&type=pdf.

Bravo, J., Hervás, R., Sánchez, I. y Crespo, A. U. (2004). Servicios por identificación en el aula ubicua. *VI Simposio Internacional de Informática Educativa (SIIE'04)* (pp. 26-27). Cáceres, España.

Cairó, B. (2006). *Fundamentos de programación. Piensa en C*. Ciudad de México: Pearson Educación.

Castro, J. I., Hernández, D. C. y Vanegas, F. A. (2013). *La deserción escolar en la carrera de ingeniería de Sistemas de la Universidad del Quindío* (trabajo de pregrado). Universidad de Quindío, Facultad de Ingeniería.

Cook, D., Augusto, J. and Jakkula, V. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, *5*(4), 277-298. Doi:10.1016/j.pmcj.2009.04.001.

Dawson-Saunders, B. y Trapp, R. (2002). *Bioestadística médica* (3.ª ed.). México: El Manual Moderno.

Díaz, F. y Hernández, G. (2002). *Estrategias docentes para un aprendizaje significativo. Una interpretación constructivista.* México, D. F.: Mc Graw Hill.

Díaz-Barriga, Á. (2013). TIC en el trabajo del aula. Impacto en la planeación didáctica. *Revista Iberoamericana de Educación Superior*, *4*(10), 3-21. Recuperado de http://ries.universia.net/index.php/ries/article/view/340.

González, E. y López, A. (en prensa). Aprendizaje metacognitivo de lenguaje, lógica y matemáticas en *smart classroom*. *Revista Electrónica de Investigación.*

González, E., López, A., Rojas, R. y Trujillo, V. (en prensa).Análisis metacognitivo de competencias adquiridas por tecnologías de la comunicación e información en universitarios. *Revista Innovación Educativa*.

Gutiérrez, F. (2005). *Teorías del proceso cognitivo*. Madrid, España: McGraw-Hill/Interamerican.

Joyanes, L. y Zahonero, I. (2005). *Programación en C. Metodología, algoritmos y estructura de datos* (2.ª ed.). Madrid, España: MacGraw Hill Interamericana.

Kori, K., Pedaste, M., Altin, H., Tõnisson, E. y Palts, T. (2016). Factors That Influence Students' Motivation to Start and to Continue Studying Information Technology in Estonia. *IEEE Transactions on Education*, *59*(4), 255–262. Doi:10.1109/TE.2016.2528889.

Kori, K., Pedaste, M., Tõnisson, E., Palts, T., Altin, H., Rantsus R. y Rüütmann, T. (2015). First-year dropout in ICT studies. *IEEE. Global Engineering Education Conference, EDUCON*, 444-452. Doi: 10.1109/EDUCON.2015.7096008.

Lázaro, A. N., Callejas, Z., Griol, D. y Durán, M. (2017). La deserción estudiantil en educación superior: S. O. S. en carreras de ingeniería Informática. *V CLABES.* Recuperado de http://revistas.utp.ac.pa/index.php/clabes/article/view/1674.

Legorreta, B. (2015). *Estrategias de aprendizaje. Fundamentos teóricos y metodológicos de la educación a distancia.* Universidad Autónoma del Estado de Hidalgo. Recuperado de http://cvonline.uaeh.edu.mx/Cursos/BV/Docentes/pdf/Tema2_estrategias.pdf.

Lozano, A. (2004). El aula inteligente: ¿hacia un nuevo paradigma educativo? [Reseña del libro *El aula inteligente: nuevas perspectivas*]. *Revista Electrónica de Investigación Educativa*, *6*(2). Recuperado de http://redie.uabc.mx/vol6no2/contenido-lozano.html.

Maheshwari, V. (2016). The concept of smart classroom. Retrieved from http://www.vkmaheshwari.com/WP/?p=2352.

Martínez, J. R. (2004). *La concepción de aprendizaje, metacognición y cambio conceptual en estudiantes universitarios de psicología* (tesis doctoral). Universidad Autónoma de Barcelona. Recuperado de www.tesisenred.net/bitstream/handle/10803/2632/Tesis_final.pdf.

O'Neil, H. F. and Abedi, J. (1996). Reliability and validity of a state metacognitive inventory: Potential for alternative assessment. *The Journal of Educational Research*, *89*(4), 234-235. Retrieved from https://doi.org/10.1080/00220671.1996.9941208.

Pinto, S. y Martínez, S. (1994). *La teoría de Jean Piaget y el aprendizaje de las ciencias. Monografía*. Colección Cuadernos del CESU, UNAM. (30), 3-110. Recuperado de https://biblat.unam.mx/es/revista/cuadernos-del-cesu-unam/.

Roa, M. (2015). *Cómo elaborar un portafolio de evidencias.* Recuperado de http://www.itmina.edu.mx/subaca/Portafolio%20de%20evidencias.pdf.

Sánchez, C., Gómez, C. y Gaviria, A. (2016): La deserción estudiantil en el programa de ingeniería de Sistemas de la Universidad de la Amazonia (2012-i a 2015-i): una lectura institucional y antropológica del asunto. *Investigación e Innovación en Ingenierías*, *4*(2), 72-118.

Sánchez, J. (2000). *Nuevas tecnologías de la información y comunicación para la construcción del aprender.* Santiago de Chile: LMA Servicios Gráficos.

Santana-Mancilla, P., Magaña, M., Rojas, J., Nieblas, J. and Salazar, A. (2013). Towards Smart Education: Ambient Intelligence in the Mexican Classrooms. *Procedia. Social and Behavioral Sciences*, (106), 3141–3148. Retrieved from https://doi.org/10.1016/J.SBSPRO.2013.12.363.

Sharma, H. (2016). Effectiveness of EDUCOMP smart classroom teaching on achievement in mathematics at elementary level. International. *Journal of Applied Research*, *3*(6), 683–687. Retrieved from http://www.allresearchjournal.com/archives/2016/vol2issue6/PartK/2-6-124-334.pdf.

Soni, T. and Dalal, N. H. (2018). *AutoGradr: Automatically grade programming assignments.* Retrieved from https://autogradr.com.

Universia España (2017). Ingeniería es el área con mayor tasa de abandono escolar en España. *Universia España*. Recuperado de http://noticias.universia.es/ciencia-tecnologia/noticia/2017/06/27/1153624/ingenieria-area-mayor-tasa-abandono-escolar-espana.html.

Universidad Autónoma del Estado de México (UAEM) (2017). *Agenda estadística 2017. México*. Recuperado de http://planeacion.uaemex.mx/docs/AE/2017/AE_2017.pdf

Vallejos, J., Jaimes, C., Aguilar, E. y Merino, M. (2012). Validez, confiabilidad y baremación del inventario de estrategias metacognitivas en estudiantes universitarios. *Revista Psicológica*, *14*(1), 9-20. Recuperado de http://sisbib.unmsm.edu.pe/BVRevistas/rev_psicologia_cv/v14_2012_1/pdf/a02v14n1.pdf.

Velazco, M. y Mosquera, F. (2015). Estrategias didácticas para el aprendizaje colaborativo. *PAIEP*. Recuperado de http://acreditacion.udistrital.edu.co/flexibilidad/estrategias_didacticas_aprendizaje_colaborativo.pdf.

Rol de Contribución	***Autor (es)***
Conceptualización	**Elvira Ivone González Jaimes**
Metodología	**Elvira Ivone González Jaimes**
Software	**Asdrúbal López Chau** **Valentín Trujillo Mora** **Rafael Rojas Hernández**
Validación	**Asdrúbal López Chau** **Rafael Rojas Hernández**
Análisis Formal	**Elvira Ivone González Jaimes** **Asdrúbal López Chau**
Investigación	**Elvira Ivone González Jaimes** **Asdrúbal López Chau**
Recursos	**Materiales de estudio software Universidad Autónoma del Estado de México**
Curación de datos	**Elvira Ivone González Jaimes** **Asdrúbal López Chau**
Escritura - Preparación del borrador original	**Elvira Ivone González Jaimes** **Asdrúbal López Chau**
Escritura - Revisión y edición	**Elvira Ivone González Jaimes** **Asdrúbal López Chau** **Valentín Trujillo Mora** **Rafael Rojas Hernández**

Visualización	**Elvira Ivone González Jaimes** **Asdrúbal López Chau**
Supervisión	**Elvira Ivone González Jaimes**
Administración de Proyectos	**Elvira Ivone González Jaimes**
Adquisición de fondos	**Elvira Ivone González Jaimes** **Asdrúbal López Chau** **Valentín Trujillo Mora** **Rafael Rojas Hernández**