Cómo citar este artículo: C. A. Arévalo-Mercado, E. L. Muñoz-Andrade, H. Cardona-Reyes and M. G. Romero-Juárez, "Applying Cognitive Load Theory and the Split Attention Effect to Learning Data Structures," in IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 18, no. 1, pp. 107-113, Feb. 2023, doi: 10.1109/RITA.2023.3250580.

Aplicación de la Teoría de Carga Cognitiva y el Efecto de Atención Dividida En La Enseñanza de Estructuras De Datos

Carlos Argelio Arévalo-Mercado, Estela Lizbeth Muñoz-Andrade, Héctor Cardona-Reyes, Martín Gabriel Romero-Juárez

***Abstract*—Learning data structures is a hard task for computer science students, given the mental effort required to simultaneously understand abstract diagrams and the dynamic manipulation of nodes and pointers using programming languages. In literature, proposed solutions to the problem focus on visualization-based artifacts, pedagogical methods, or a combination of both. The present study is framed within the cognitive learning paradigm and describes the design and testing of a linked list visualization software tool, based on the Split Attention effect of Cognitive Load Theory. The study was carried out at the Autonomous University of Aguascalientes (UAA), Mexico. In the learning effectiveness test, significant results (p=0.000) are reported for the participants of the experimental group (n=35), using the nonparametric Wilcoxon test, with a quasi-experimental pre-posttest design. It is discussed that the spatial and temporal integration of linked list node diagrams and the corresponding worked example code for the implementation of its basic operations can benefit students with learning gaps in previous introductory programming courses. It is also reported that the control group (n=36) had gains through traditional learning (p=0.022), although this group started from a higher prior academic performance. We propose to extend the Split Attention Tool to include a wider range of data structures and to replicate the study with randomized experimental designs.**

***Index Terms*—Cognitive Load Theory, Split Attention Effect, Data Structures, Instructional Design, Programming Education, Digital Competence, Software Engineering, Latin America.**

I. Introduction

El aprendizaje de la programación para los estudiantes que inician en carreras afines a las ciencias computacionales, continúa siendo un tema relevante de investigación tanto por su complejidad inherente relacionada con el manejo de diferentes niveles de abstracción [1], la dificultad sintáctica de los lenguajes de programación [2] y la adquisición de modelos mentales sobre el comportamiento de las estructuras de control necesarias para la escritura de programas [3], [4]. Otros factores reportados son la falta de planes de programación previos para la resolución de problemas y la autoeficacia [5], [6].

Estudios recientes han intentado dimensionar con mayor precisión la problemática de la reprobación en materias introductorias de programación [7], [8], en los que se estiman porcentajes en las universidades los Estados Unidos entre el 28% y el 33% y se concluye que estas cifras ya no son “alarmantemente altas”, aunque se concede que aún hay espacio para la mejora. En Latinoamérica, es difícil localizar estudios multinacionales de esta naturaleza, encontrándose solamente cifras a nivel país o Universidad [9]. En la Universidad Autónoma de Aguascalientes (UAA), donde el presente estudio fue llevado a cabo, se reportan indicadores de reprobación entre el 39% y el 45% entre los años 2016 y 2019.

En los cursos de Estructuras de Datos (CS3), que comúnmente son precedidos por al menos dos cursos de introducción a la programación, la complejidad conceptual es aún mayor para los estudiantes de Informática, ya que al inicio de este curso deben dominar la algoritmia y las estructuras de datos básicas de los dos cursos anteriores para poder codificar estructuras de datos como pilas, colas, listas y árboles, con diversas variantes para cada una de ellas. En el ámbito profesional, estas estructuras de datos son la base para crear algoritmos de optimización, recursividad, búsqueda y programación paralela, e incluso tecnologías más avanzadas como Blockchain, por lo que su dominio es importante para la implementación de soluciones eficientes basadas en software.

En la literatura, las propuestas para ayudar con este problema de aprendizaje se pueden encontrar en dos grandes categorías: herramientas de visualización [10]-[12] y soluciones basadas en enfoques pedagógicos o metodológicos, como el aprendizaje activo [13] o la programación por parejas [14]. Algunas de ellas incluyen una combinación de ambos enfoques [15].

Este trabajo se inscribe en esta última categoría y presenta un estudio de caso realizado con estudiantes de la Universidad Autónoma de Aguascalientes (UAA), México. Se presenta una

Carlos Argelio Arévalo Mercado is with the Autonomous University of Aguascalientes (UAA), Aguascalientes, México (e-mail: carlos.arevalo@edu.uaa.mx). ORCID (0000-0002-8349-7985)

Estela Lizbeth Muñoz-Andrade is with the Autonomous University of Aguascalientes (UAA), Aguascalientes, México (e-mail: lizbeth.munoz@edu.uaa.mx). ORCID (0000-0003-4182-5044)

Héctor Cardona-Reyes is with the CONACYT, CIMAT Zacatecas, Mexico (Corresponding author: e-mail: hector.cardona@cimat.mx). ORCID (0000−0002−9626−6254)

Martín Gabriel Romero-Juárez is with the Autonomous University of Aguascalientes, Aguascalientes, Mexico (e-mail: mgabriel.rj94@gmail.com)

herramienta de Visualización de Estructuras de Datos, incluyendo el código de implementación como ejemplos trabajados, y un diseño de interfaz de usuario basado en el Efecto de Atención Dividida de la Teoría de la Carga Cognitiva.

En este estudio de caso, se utiliza la prueba no paramétrica de Wilcoxon, con un diseño cuasiexperimental de prueba pre-post. Por último, se discute que la integración espacial y temporal de los diagramas de nodos de listas enlazadas y el correspondiente código de ejemplo trabajado para la implementación de sus operaciones básicas pueden beneficiar a los estudiantes con menor rendimiento de aprendizaje en los cursos previos de introducción a la programación.

Este trabajo se compone de siete secciones. La siguiente sección de antecedentes discute las características teóricas del paradigma de aprendizaje cognitivo y, en particular, la teoría de la carga cognitiva y el efecto de atención dividida. En la sección tres, dedicada a los trabajos relacionados, se describen los resultados de tres trabajos de la literatura sobre la visualización de la estructura de los datos y los enfoques didácticos. La sección cuatro describe los principios de diseño instructivo de la herramienta de aprendizaje y los elementos del diseño cuasiexperimental utilizado para medir el aprendizaje de las listas enlazadas. La sección cinco de resultados presenta los datos cuantitativos (estadística descriptiva y prueba no paramétrica de Wilcoxon pre-post). La discusión sobre las diferencias en el rendimiento de los grupos participantes y la pertinencia del uso de la Herramienta de Atención Dividida para los estudiantes con menor rendimiento académico, así como las diferencias y similitudes con otros trabajos se presentan en la sección seis. Finalmente, las conclusiones de este trabajo y el trabajo futuro de esta investigación se presentan en la sección siete.

II. Estado del Arte

El paradigma cognitivo parte del supuesto de que el aprendizaje es la adquisición activa de esquemas mentales adecuados [16], mediante la práctica deliberada y constante. Supone que, en las etapas iniciales de aprendizaje, el estudiante debe ser asistido mediante diseños instruccionales que minimicen la carga cognitiva en la memoria de corto plazo, mientras que progresivamente se disminuye el andamiaje instruccional cuando éste adquiere mayor conocimiento y habilidad [17] mediante un proceso mental denominado automatización.

En este contexto, la Teoría de Carga Cognitiva [18] propone distintos diseños instruccionales llamados “efectos” para disminuir la carga mental y facilitar el aprendizaje. De éstos, el efecto del ejemplo resuelto es el más estudiado y el que cuenta con mayor evidencia empírica sobre su efectividad en la transferencia de conocimiento [19], [20]. Otro de estos efectos es el llamado efecto de atención dividida, el cual señala que, si el aprendiz debe integrar mentalmente dos fuentes de información física o temporalmente dispersas para comprender una solución o ejemplo resuelto, este proceso genera una carga cognitiva innecesaria que dificulta el aprendizaje. El efecto de atención dividida se puede prevenir integrando física o temporalmente el diagrama y los enunciados de la solución, haciendo que la integración mental sea mínima y se reestablezcan los efectos positivos de los ejemplos resueltos [21].

En el área de la programación, la enseñanza de las estructuras de datos se basa fuertemente en la utilización de diagramas para representar combinaciones de formas abstractas tales como nodos y apuntadores, y estos diagramas van acompañados del código que implementa las operaciones que se realizan sobre ellos, tales como la creación, inserción, borrado, recorrido y búsqueda.

Generalmente, la representación diagramática y la implementación en código suelen estar divididos tanto espacial como temporalmente, generando el efecto de atención dividida descrito por la Teoría de Carga Cognitiva.

Con base en lo anterior, en el presente estudio se describe el diseño y prueba de la efectividad para el aprendizaje de una herramienta para la enseñanza de estructuras de datos basada en el efecto de atención dividida.

III. Trabajo Relacionado

En la literatura se pueden encontrar varios trabajos relacionados con el uso de herramientas de visualización como apoyo a la enseñanza de estructuras de datos. Por ejemplo, [12] reporta un estudio piloto con resultados positivos de usabilidad de una herramienta de aprendizaje llamada 'DstBlocks', basada en el paradigma pedagógico constructivista. En este estudio, la medición del constructo de aprendizaje no se midió con problemas de transferencia cercana (near-transfer), sino mediante preguntas tipo encuesta.

La transferencia cercana se produce cuando la nueva situación de aprendizaje es similar a una situación anterior, y sólo difiere ligeramente de ella, mientras que la transferencia lejana se produce cuando la nueva situación de aprendizaje tiene patrones diferentes a los de las anteriores [22].

En [10] se describe una herramienta para la visualización de estructuras de datos y código llamada 'Willow', que utiliza Python como lenguaje de codificación de ejemplo. Aunque no se informa de que su diseño esté basado en un enfoque pedagógico o de aprendizaje, la flexibilidad proporcionada por las capacidades de edición y visualización de código en varios idiomas es prometedora. Los resultados de un estudio cualitativo con 7 profesores participantes son positivos. Los autores reconocen que la herramienta necesita una mayor validación de su eficacia en el aula.

En [23] se describe un diseño instruccional para un curso de Estructuras de Datos y Programación Paralela, centrado en el aprendizaje del pensamiento convergente y divergente [24], (que se refiere a "*encontrar diferentes soluciones para problemas similares (divergencia) y seleccionar una solución apropiada dadas las restricciones y suposiciones sobre un problema (convergencia))*", con resultados positivos en la adquisición de este tipo de habilidades, pero no se reportan resultados en la adquisición de habilidades de resolución de problemas a través de problemas de tipo transferencia cercana.

En [13] se reporta el uso del enfoque pedagógico conocido como 'Aprendizaje Activo' [25] para la enseñanza de la programación paralela y estructuras de datos. Se utiliza el componente OpenMP para C++, como capa adicional de abstracción para reducir la complejidad de la sintaxis de los algoritmos de paralelización. Para la visualización se utilizaron animaciones de PowerPoint. Se reportan resultados cualitativos positivos sobre el uso del enfoque y mejoras en el aprendizaje de los conceptos de paralelización y estructuras de datos, pero se sugiere que se apliquen posteriores estudios para validar el enfoque de aprendizaje activo en este contexto.

En [26] se describe una herramienta de visualización de estructuras de datos con amplios detalles técnicos sobre su implementación e interfaz de usuario. Utiliza Java para permitir escribir algoritmos y un analizador léxico para generar tokens que a su vez son utilizados por un analizador sintáctico para generar código Java.

Esta herramienta tiene similitudes técnicas con la presentada en este estudio, pero no incluye una teoría de diseño instruccional subyacente ni mediciones cuantitativas del aprendizaje.

Como puede deducirse de los métodos y resultados de los trabajos anteriores relacionados, se hace hincapié en la visualización novedosa y en los métodos pedagógicos (en su mayoría procedentes del paradigma del aprendizaje activo/constructivista) y se informa de los comentarios positivos del uso por parte de los estudiantes y del profesorado, pero carecen de datos cuantitativos sobre la transferencia del aprendizaje. Es decir, el concepto de transferencia en el aprendizaje suele referirse a la capacidad de los estudiantes para recordar conocimientos y habilidades previos y aplicarlos en nuevas situaciones de aprendizaje.

En este contexto, el presente estudio pretende ofrecer resultados cuantitativos de cuasi transferencia, dentro de un paradigma de aprendizaje cognitivo, en contraposición a un enfoque constructivista de aprendizaje colaborativo.

IV. Método

El método aplicado para este estudio se dividió en etapas: En primer lugar, el diseño y desarrollo de la herramienta de apoyo al aprendizaje de estructuras de datos, siguiendo estrechamente las pautas instructivas del Efecto de Atención Dividida de la Teoría de la Carga Cognitiva, y en segundo lugar la comprobación de la eficacia del aprendizaje mediante un diseño controlado y cuasiexperimental centrado en la adquisición de habilidades casi transferibles. Es decir, la resolución de nuevos ejercicios de estructuras de datos de una estructura de problema similar.

A. *Diseño de la Herramienta*

La herramienta se desarrolló con JavaScript y JQuery. La visualización de los diagramas de listas enlazadas se consiguió utilizando la etiqueta 'Canvas' de JavaScript, que permite dibujar gráficos o animaciones sencillas.

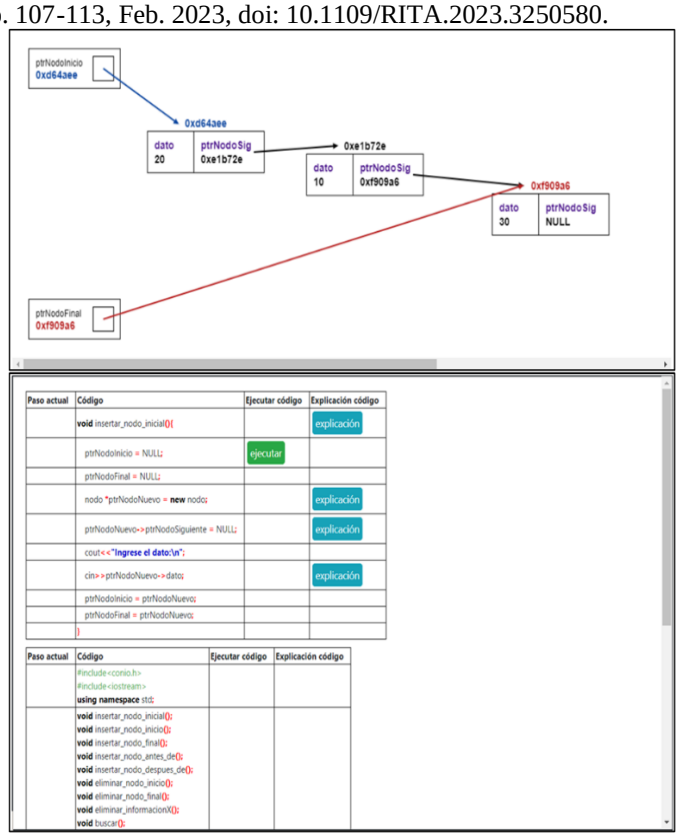


Fig. 1. Interfaz gráfica e interactiva de la herramienta basada en el efecto de atención dividida.

Fig. 2. Botón de explicación del código de ejemplo C++ para crear un nodo para listas enlazadas.

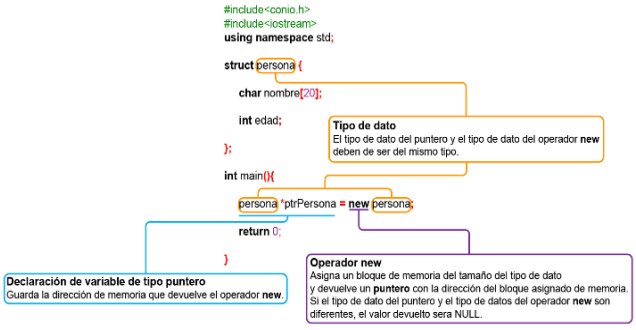


Fig. 3. Ayuda contextual de sintaxis de C++ para la creación de apuntadores y nodos.

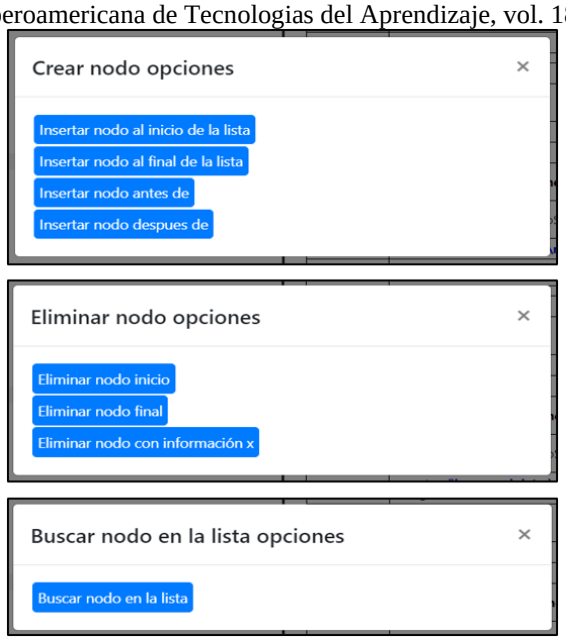


Fig. 4. Operaciones contextuales aplicables a los nodos de la estructura de datos.

La pantalla principal de la herramienta está dividida en dos secciones (Figura 1), una con la representación esquemática y dinámica de la creación de nodos y punteros, que puede ser manipulada con la funcionalidad de arrastre, y otra, también interactiva, que muestra contextualmente uno o varios ejemplos trabajados (dependiendo de la acción realizada sobre los nodos) en código C++ (ver Figuras 1, 2 y 4).

Para reducir la complejidad y mejorar la comprensión de la sintaxis, cada línea de código dispone de un botón de explicación para mostrar una ayuda contextual sobre la estructura sintáctica de C++, que puede mostrarse opcionalmente (Figuras 2 y 3).

El alumno recorre y ejecuta el código línea por línea, que se sincroniza temporalmente con la visualización y el efecto esquemático de cada acción. Es decir, como sobre una estructura de datos se pueden ejecutar diversas operaciones (insertar, borrar, buscar, modificar, etc.), la herramienta permite visualizar gráficamente la implementación equivalente en código de la operación seleccionada por el alumno (Figura 4).

Por motivos de prueba y debido al gran número de estructuras de datos que existen (pilas, colas, listas, árboles, etc.), la primera versión de la herramienta sólo incluye el comportamiento y las operaciones de las listas unidireccionales.

B. Diseño Experimental

La población objetivo fueron los estudiantes de Ciencias de la Computación que estudian Estructuras de Datos. Se optó por un diseño cuasiexperimental, no aleatorio, pre-post.

El pre-test consistió en la realización de 5 ítems del 2º examen parcial, asociados a problemas de listas enlazadas, donde cada ítem correcto tenía un valor de 2 puntos, asignándose puntos a las respuestas parcialmente correctas. El post-test consistió en 5 problemas isomórficos del tema de listas sencillamente enlazadas y fueron aplicados como parte del examen final del semestre de ambos grupos. Los problemas fueron tomados del banco de preguntas estandarizadas de la academia de programación.

Los siguientes ejemplos ilustran el formato de los ejercicios del pre-test y del post-test correspondiente.

Escribe una función que, dadas dos listas ordenadas de menor a mayor, devuelva otra lista con todos los elementos de las dos listas originales y ordenada de menor a mayor. (pre-test)

Escribe una función que, dada una lista L, devuelva otra lista R que contenga los elementos repetidos de L. Por ejemplo, si L almacena los valores 5, 2, 7, 2, 5, 5, 1, se debe construir una lista R con los valores 5, 2. Si no hay elementos repetidos en L, R será la lista vacía. (post-test)

Como se mencionó anteriormente, la estructura de los 5 ejercicios se diseñó para que fueran isomórficos y de naturaleza near-transfer. Es decir, tenían descripciones diferentes pero un nivel de dificultad similar que necesitaba esquemas similares para resolver el problema, para dar cuenta de una mejor medición del aprendizaje después del tratamiento con la herramienta de atención dividida.

Los participantes fueron de dos grupos previamente conformados del 3er semestre de la carrera de Ingeniería en Ciencias de la Computación de la Universidad Autónoma de Aguascalientes (UAA), México. Ambos grupos tuvieron el mismo instructor durante el semestre, pero diferentes profesores en cursos de programación anteriores.

El grupo "A" fue seleccionado al azar como grupo de control (n=36) y el grupo "C" fue el grupo experimental (n=35). El grupo experimental recibió una sesión de laboratorio de una hora de duración para aprender a utilizar la herramienta y para darles una cuenta de acceso.

A ambos grupos se les proporcionaron 10 problemas sobre el tema de listas simplemente ligadas, distribuidos en 3 niveles de dificultad. Al grupo experimental se le permitió utilizar la herramienta de atención dividida como apoyo para resolver los ejercicios. El grupo de control pudo utilizar sus apuntes y el material didáctico proporcionado por el profesor.

Dos ejemplos de estos ejercicios de nivel de dificultad bajo y medio, respectivamente, son los siguientes:

Ejercicio de ejemplo 1 (dificultad baja):

"Implementar una función que devuelva la suma de los números introducidos en los nodos de la lista simplemente ligada."

Ejercicio de ejemplo 2 (dificultad media)

"Implementar un procedimiento para insertar un dato (int) en orden ascendente en una lista enlazada. Es decir, el nodo que representa el dato debe ser insertado en una posición tal que al recorrer la lista los nodos se recorran de menor a mayor respecto del dato".

La aplicación de la prueba de práctica del tratamiento fue simultánea y supervisada in situ para ambos grupos, que además se mantuvieron separados para evitar que se compartieran las respuestas correctas. La duración de esta prueba fue de 2 horas.

La variable dependiente se definió como el aprendizaje de las listas simplemente ligadas. La variable independiente se definió como el método de aprendizaje con dos niveles: tradicional y experimental.

V. Resultados

Los resultados de la estadística descriptiva se muestran en la Tabla I. La desviación estándar corresponde al valor entre paréntesis. Se observa que las medias de ambos grupos tuvieron incrementos en las post pruebas. El grupo experimental muestra un incremento de 3.05 puntos en la escala de evaluación utilizada (de 0 a 10) y el grupo de control muestra un incremento de 0.89 puntos.

El valor asociado a 'Conocimientos previos' se obtuvo a partir de la calificación promedio de cada grupo en el curso curricular anterior de programación (Programación I, o CS2, del segundo semestre).

El lapso de tiempo entre el tratamiento y el post-test fue de 2 semanas durante noviembre de 2019.

En la Tabla 2 se muestran las modas y el valor inmediatamente inferior de la distribución de frecuencias de cada grupo, en donde para el grupo experimental la moda del pre-test fue de 6, con 14 observaciones y el valor inmediatamente inferior fue de 5, con 9 observaciones. No hubo observaciones de 10 en pre-test del grupo experimental. Para el mismo grupo experimental, la moda del post-test fue de 10, con 18 observaciones y el valor inmediatamente inferior en cuanto a frecuencias fue de 9, con 7 observaciones.

Para el grupo experimental, la moda pre-test fue de 10, con 14 observaciones y la moda del post-test fue de 10, con 25 observaciones. El comportamiento de las frecuencias en las mediciones pre y post de ambos grupos puede visualizarse en los histogramas de las Figuras 5 y 6.

Los histogramas Pre y Post del grupo experimental permiten observar que la mayoría de los resultados del pre-test se acumulan en los rangos 5 y 6 y que corresponden al 65% del total, en tanto que en la medición posterior al tratamiento se observa que la mayoría de los resultados están en el rango de 9 y 10 (71%).

En los histogramas del grupo de control, puede observarse que los rangos de calificaciones más frecuentes en las mediciones Pre y Post, corresponden a 9 y 10, con la diferencia de que en la medición Post, ya no se registraron observaciones en los rangos menores de 3, 4, 5 y 6. Acumulándose además el 69% de las observaciones en el rango de 10.

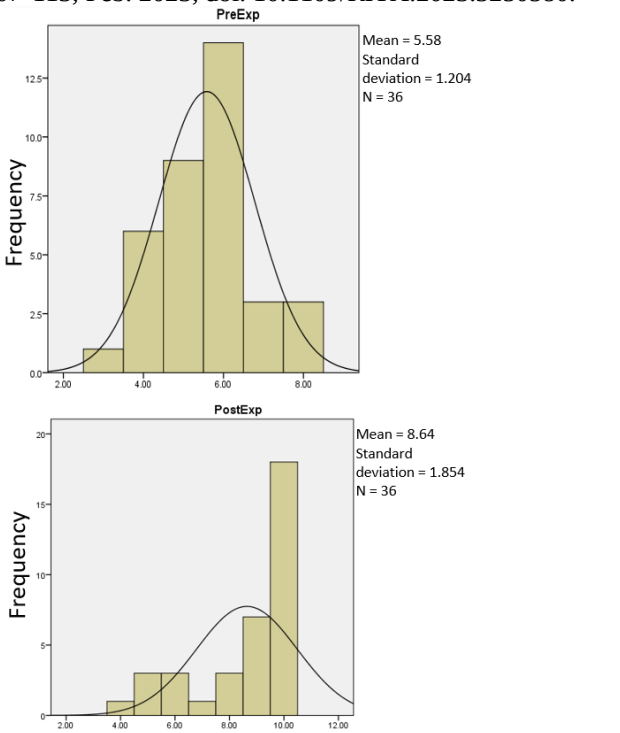


Fig. 5. Histogramas de observaciones pre y post del grupo experimental.

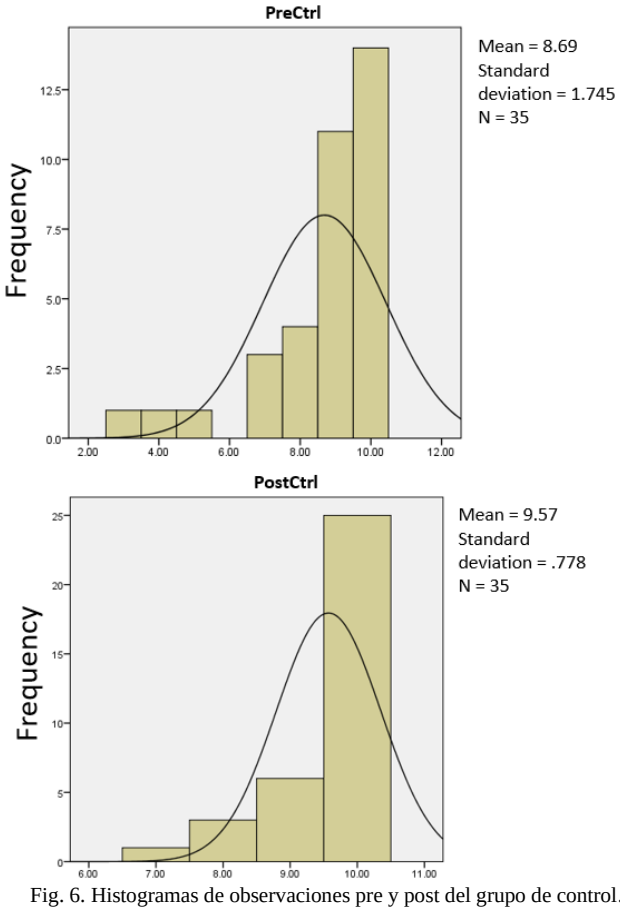


Fig. 6. Histogramas de observaciones pre y post del grupo de control.

TABLA I
MEDIA Y DESVIACIÓN ESTÁNDAR DE GRUPOS EXPERIMENTAL Y DE CONTROL EN MEDICIONES 'CONOCIMIENTO PREVIO', 'PRE-TEST' Y 'POST-TEST'

	Grupo Experimental	Grupo de Control
Conoc. Previo	6.29 (2.47)	8.04 (1.69)
Pre – test	5.58 (1.20)	8.68 (1.74)
Post - test	8.63 (1.85)	9.57 (0.77)

TABLA II
MODAS Y SUS FRECUENCIAS, GRUPOS EXPERIMENTAL Y DE CONTROL EN MEDICIONES 'PRE' Y 'POST'

	Grupo Experimental	Grupo de Control
Pre – test	6 (14), 5 (9)	10 (14), 9 (11)
Post - test	10 (18), 9 (7)	10 (25), 9 (6)

TABLA III
RESULTADOS PRUEBA NO PARAMÉTRICA DE WILCOXON DE RANGOS CON SIGNO PARA MUESTRAS RELACIONADAS.

Hipótesis	Sig.	Decision
La mediana de las diferencias entre las observaciones Pre y Post del grupo experimental es igual a 0	0.000	Rechazar la hipótesis nula
La mediana de las diferencias entre las observaciones Pre y Post del grupo de control es igual a 0	0.022	Rechazar la hipótesis nula

Las pruebas de normalidad de Kolmogorov y Shapiro-Wilk para el Pre y Post test de ambos grupos fueron ambas rechazadas (p=0.000), por lo que una prueba t de muestras relacionadas no era aplicable a los resultados. Por tanto, se aplicó la prueba no paramétrica de Wilcoxon de comparación de medianas para muestras relacionadas (Ver Tabla III), resultando ambas pruebas en el rechazo de la hipótesis nula de igualdad (p=0.000, y p=0.022), lo que señala que para ambos grupos la diferencia estadística entre las mediciones pre y post es significativa.

VI. DISCUSIÓN

El grupo de control mostró un mejor rendimiento al inicio y al final del estudio, reflejado en los valores de las medias inicial y final (8.68 y 9.57) e incluso una mejora estadísticamente significativa (p=0.022), de acuerdo con la prueba de Wilcoxon. Estos resultados son consistentes con el rendimiento académico previo (8.04) de este grupo, ya que el estudio se realizó con estudiantes con un año de experiencia en programación y este grupo en particular contó con un mejor aprendizaje previo que el grupo experimental y el repaso de los temas mediante los 10 ejercicios de práctica asignados durante el tratamiento pudo influir en el reforzamiento de esquemas y en el resultado positivo de la medición 'post'.

Por otro lado, en el grupo experimental se observa que a pesar de iniciar con un rendimiento académico previo menor (6.29) y también una medición menor del Pre-test (5.58), éste tuvo una ganancia importante en el post-test, siendo ésta de 3 puntos, y también una diferencia de medias estadísticamente significativa (p=0.000) en la prueba de Wilcoxon.

Creemos que este grupo de menor rendimiento posiblemente se benefició del efecto de atención dividida, a través de las características de visualización, y de la ejecución simultánea de código fomentando la creación y/o corrección de esquemas mentales en la memoria a largo plazo, asociados con el comportamiento de las listas enlazadas individualmente y, por lo tanto, en el aprendizaje de near-transfer.

Dados los resultados, creemos que tanto la herramienta aquí descrita, como los principios de diseño instruccional de la Teoría de la Carga Cognitiva propuestos por [27], específicamente el efecto de Atención Dividida, pueden ser una ayuda valiosa para los estudiantes con lagunas de aprendizaje de programación previas para el aprendizaje de estructuras de datos. Por otro lado, dados los resultados del grupo de control, también observamos que, para los estudiantes de alto rendimiento, la práctica constante a través de los ejercicios tradicionales seguía siendo una estrategia de aprendizaje eficaz.

VII. LIMITACIONES Y TRABAJO FUTURO

El efecto de la variable asociada al rendimiento académico previo no fue controlado, pero si reportado, lo que se corrobora en la media del pre-test del grupo de control. La posibilidad de aprendizaje colaborativo durante el tratamiento no fue controlada y los alumnos de ambos grupos pudieron compartir opiniones y resultados de los ejercicios de práctica.

Dado que el diseño de este estudio fue cuasiexperimental, los resultados no son generalizables, requiriendo futuras réplicas aleatorias que consideren el rendimiento académico previo. En el contexto de la Teoría de la Carga Cognitiva, se describen otros efectos que podrían considerar el nivel inicial de pericia de los estudiantes, para quienes el efecto del ejemplo resuelto ya no es efectivo. Tal es el caso del efecto "Guidance Fading" [21], que puede ser explorado en futuros estudios.

Otro factor por considerar está relacionado con la persistencia del efecto de la herramienta entre el final del tratamiento y el post-test, que fue de dos semanas. Es decir, aunque se verificó en los registros de acceso a la herramienta que los alumnos siguieron utilizándola regularmente, el grupo experimental pudo haber utilizado otras fuentes adicionales de aprendizaje, diluyendo los efectos del tratamiento.

En este escenario, para evitar la posible dilución del efecto en futuros estudios aleatorios, se puede acortar la duración del estudio y pedir explícitamente a los participantes que eviten fuentes adicionales de aprendizaje de estructuras de datos durante el experimento.

Por último, la versión actual de la herramienta se está ampliando para soportar el aprendizaje de otras estructuras de datos, como listas circulares, pilas, colas y árboles.

AGRADECIMIENTOS

Los autores agradecen el apoyo de la Academia de Programación del Departamento de Sistemas de Electrónicos de la UAA, especialmente al Dr. Eduardo Serna Pérez, por su colaboración estrecha con los alumnos del curso de estructuras de datos"

REFERENCIAS

[1] A. Benander, B. Benander, and J. Sang, "Factors related to the difficulty of learning to program in Java - An empirical study of non-novice programmers," *Inf. Softw. Technol.*, vol. 46, no. 2, pp. 99–107, 2004, doi: 10.1016/S0950-5849(03)00112-5.

[2] M. F. Yiğit and M. Başer, "Learning Difficulties and Use of Visual Technologies in Learning to Program," *Particip. Educ. Res.*, vol. spi15, no. 2, pp. 27–34, Nov. 2015, doi: 10.17275/per.15.spi.2.4.

[3] I. Milne and G. Rowe, "Difficulties in Learning and Teaching Programming—Views of Students and Tutors," *Educ. Inf. Technol.*, vol. 7, no. 1, pp. 55–66, 2002, [Online]. Available: http://files/659/Difficulties in Learning and Teaching programmng.pdf.

[4] T. Jenkins, "On the difficulty of learning to program," 2002, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.596.9994&rep=rep1&type=pdf.

[5] C. Argelio, A. Mercado, E. Lizbeth, M. Andrade, J. Manuel, and G. Reynoso, "El efecto de la autoeficacia y el trabajo colaborativo en estudiantes novatos de programación The effect of self-efficacy and peer collaboration in novice programming students," *Investig. Y Cienc. LA Univ. AUTÓNOMA AGUASCALIENTES*, 2018.

[6] C. C. Tseng, P. Y. Chao, and K. R. Lai, "An Analysis of Goal Orientation Pattern and Self-Efficacy for Explanation of Programming Plans," in *2015 IEEE 15th International Conference on Advanced Learning Technologies*, 2015, pp. 76–77, doi: 10.1109/ICALT.2015.93.

[7] C. Watson and F. W. B. Li, "Failure rates in introductory programming revisited," in *ITICSE 2014 - Proceedings of the 2014 Innovation and Technology in Computer Science Education Conference*, 2014, pp. 39–44, doi: 10.1145/2591708.2591749.

[8] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming - 12 years later," *ACM Inroads*, vol. 10, no. 2, pp. 30–35, 2019, doi: 10.1145/3324888.

[9] J. Juáres, M. López, and Y. Villareal, "Estrategias para Reducir el Indice de Reprobación en Fundamentos de Programación de Sistemas Computacionales del I . T . Mexicali," *Rev. Gestión Empres. y Sustentabilidad*, vol. 2, no. 1, pp. 25–41, 2016, Accessed: Mar. 23, 2021. [Online]. Available: https://ideas.repec.org/a/msn/rgjrnl/v2y2016i1p25-41.html.

[10] P. Moraes and L. Teixeira, "Willow: A tool for interactive programming visualization to help in the data structures and algorithms teaching-learning process," *ACM Int. Conf. Proceeding Ser.*, pp. 553–558, 2019, doi: 10.1145/3350768.3351303.

[11] J. A. Jimnez-Murillo, O. Ortz-Ortz, P. Jimnez-Hernndez, L. N. Alvarado-Zamora, and E. M. Jimnez-Hernndez, "The teaching-learning of relations in discrete mathematics using software as a support," *Proc. - 2017 5th Int. Conf. Softw. Eng. Res. Innov. CONISOFT 2017*, vol. 2018-Janua, pp. 213–217, 2018, doi: 10.1109/CONISOFT.2017.00033.

[12] D. F. Almanza-Cortés, M. F. Del Toro-Salazar, R. A. Urrego-Arias, P. G. Feijóo-García, and F. D. De la Rosa-Rosero, "Scaffolded block-based instructional tool for linear data structures: A constructivist design to ease data structures' understanding," *Int. J. Emerg. Technol. Learn.*, vol. 14, no. 10, pp. 161–179, 2019, doi: 10.3991/ijet.v14i10.10051.

[13] M. A. Kuhail, S. Cook, J. W. Neustrom, and P. Rao, "Teaching parallel programming with active learning," *Proc. - 2018 IEEE 32nd Int. Parallel Distrib. Process. Symp. Work. IPDPSW 2018*, pp. 369–376, 2018, doi: 10.1109/IPDPSW.2018.00069.

[14] Y. F. Yang, C. I. Lee, and C. K. Chang, "Learning motivation and retention effects of pair programming in data structures courses," *Educ. Inf.*, vol. 32, no. 3, pp. 249–267, 2016, doi: 10.3233/EFI-160976.

[15] R. C. M. Correia, R. E. Garcia, C. Olivete, A. C. Brandi, and G. P. Cardim, "A methodological approach to use technological support on teaching and learning data structures," *Proc. - Front. Educ. Conf. FIE*, vol. 2015-Febru, no. February, 2015, doi: 10.1109/FIE.2014.7043992.

[16] C. C. Carbon and S. Albrecht, "Bartlett's schema theory: The unreplicated 'portrait d'homme' series from 1932," *Q. J. Exp. Psychol.*, vol. 65, no. 11, pp. 2258–2270, Nov. 2012, doi: 10.1080/17470218.2012.696121.

[17] A. Renkl, R. K. Atkinson, U. H. Maier, and R. Staley, "From example study to problem solving: Smooth transitions help learning," *J. Exp. Educ.*, vol. 70, no. 4, pp. 293–315, 2002.

[18] J. Sweller, "Cognitive Load Theory and Computer Science Education," pp. 1–1, 2016, doi: 10.1145/2839509.2844549.

[19] J. Sweller and G. A. Cooper, "The use of worked examples as a substitute for problem solving in learning algebra.," *Cogn. Instr.*, vol. 2, no. 1, pp. 59–89, 1985, doi: 10.1207/s1532690xci0201_3.

[20] R. Zhi, T. Price, S. Marwan, A. Milliken, T. Barnes, and M. Chi, "Exploring the impact of worked examples in a novice programming environment," *SIGCSE 2019 - Proc. 50th ACM Tech. Symp. Comput. Sci. Educ.*, 2019, doi: 10.1145/3287324.3287385.

[21] J. Sweller, J. J. G. van Merriënboer, and F. Paas, "Cognitive Architecture and Instructional Design: 20 Years Later," *Educational Psychology Review*, vol. 31, no. 2. Springer New York LLC, pp. 261–292, Jun. 15, 2019, doi: 10.1007/s10648-019-09465-5.

[22] Y. J. Dori and I. Sasson, *"A three-attribute transfer skills framework-part I: Establishing the model and its relation to chemical education,"* Chem. Educ. Res. Pract., vol. 14, no. 4, pp. 363–375, 2013, doi: 10.1039/c3rp20093k.

[23] P. D. Reddy, S. Iyer, and M. Sasikumar, "Teaching and learning of divergent and convergent thinking through open-problem solving in a data structures course," *Proc. - 2016 Int. Conf. Learn. Teach. Comput. Eng. LaTiCE 2016*, pp. 178–185, 2016, doi: 10.1109/LaTiCE.2016.13.

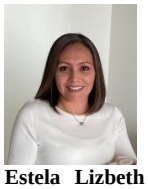
[24] M. Basadur, M. Wakabayashi, and G. B. Graen, "Individual Problem-Solving Styles and Attitudes Toward Divergent Thinking Before and After Training," *Creat. Res. J.*, vol. 3, no. 1, pp. 22–32, 1990, doi: 10.1080/10400419009534331.

[25] N. Apkarian, C. Henderson, M. Stains, J. Raker, E. Johnson, and M. Dancy, "What really impacts the use of active learning in undergraduate STEM education? Results from a national survey of chemistry, mathematics, and physics instructors," *PLoS One*, vol. 16, no. 2 February, Feb. 2021, doi: 10.1371/JOURNAL.PONE.0247544.

[26] T. Chen and T. Sobh, *"A tool for data structure visualization and user-defined algorithm animation,"* Proc. - Front. Educ. Conf., vol. 1, p. T1D/2-T1D/7, 2001, doi: 10.1109/FIE.2001.963845.

[27] J. Sweller, P. Ayres, and S. Kalyuga, *Cognitive Load Theory*. New York, NY: Springer New York, 2011.

Carlos Argelio Arévalo-Mercado. Es profesor investigador de tiempo completo en el Departamento de Sistemas de Información de la Universidad Autónoma de Aguascalientes (UAA), México. Fue jefe del mismo departamento entre 2014 y 2020. Es doctor en Ciencias Exactas y Sistemas de Información por la misma universidad. Sus áreas de investigación se centran en el diseño y desarrollo de tecnologías de aprendizaje y métodos didácticos basados teorías de las ciencias cognitivas, particularmente en el aprendizaje de la programación. Es colaborador frecuente de la Universidad de Ciencias aplicadas de Tampere (TAMK), Finlandia, en proyectos multinacionales con financiamiento Erasmus+.



Estela Lizbeth Muñoz-Andrade Es profesora investigadora de tiempo completo adscrita al Departamento de Sistemas Electrónicos de la Universidad Autónoma de Aguascalientes, México. Doctora en Ciencias Exactas y Sistemas de Información por la misma Universidad. Coordinadora de la Academia de Software de Base y Programación de Sistemas. Sus áreas de investigación se enfocan al desarrollo de software educativo para el aprendizaje de la programación.

Héctor Cardona-Reyes Es investigador CONACYT asignado al Centro de investigación en matemáticas en Zacatecas, México. Recibió el grado de Dr. En ciencias de la computación en la Universidad Juárez Autónoma Tabasco, México. Sus temas de investigación incluyen Interacción Humano-Computadora, entornos interactivos aplicados a salud y educación, ingeniería web, diseño de videojuegos y realidad virtual. Adicionalmente, es miembro nivel 1 del Sistema Nacional de Investigadores de CONACYT en México.

Martín Gabriel Romero-Juárez. Es Máster en Informática y Tecnologías Computacionales de la Universidad Autónoma de Aguascalientes. Su línea de investigación se enfoca en el diseño y desarrollo de objetos de aprendizaje guiados por teorías de las ciencias cognitivas para la enseñanza de la programación.