Conference Paper

Methodological Process for the Teaching of Computer Programming based on Computational Thinking: Case Study

Emmanuel Lopez-Neri and Estela Torres-Santoyo

Universidad del Valle de México, CIIDETEC-UVM, Campus Guadalajara Sur,45601 Santa María Tequepexpan, Jalisco, México

Abstract

In academia, it is common to identify the problem solving process based on computational thinking, as the traditional method of programming teaching. However, students would first have to develop the four types of thinking involved in this process, in order to develop successfully the programming skills. Therefore is required from the beginning of the learning process a method that provides students with a contextualization, allowing the configuration of their own language, which propitiates the development of analytical thinking for the construction of solutions for increasingly complex problems. This paper describes a methodological process of computer programming teaching based on the computational thinking process, by integrating components that promote the development of analytical thinking. Finally, we present a case study with STEM undergraduate students as participants.

Keywords: Teaching programming, analytical thinking, methodological process, computational thinking, decomposition.

Corresponding Author:
Emmanuel Lopez-Neri
emmanuel.lopezne@uvmnet.edu

Received: 28 July 2017
Accepted: 5 September 2017
Published: 30 January 2018

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the SIIPRIN Conference Committee.

1. Introducción

La programación ha sido en los últimos años una habilidad deseada por las industrias, por los gobiernos y en general por la ciencia [1, 2], sin embargo, el interés por carreras afines a esta habilidad ha ido disminuyendo drásticamente [3], y la tasa de deserción ha ido en incremento [4]. Uno de los principales factores es la complejidad del aprendizaje de la programación, el cual es un elemento fundamental en estas carreras. Por lo que se han buscado distintas estrategias de enseñanza que permitan el aprendizaje de la programación para la resolución de problemas.

En general el método usado para la enseñanza de la programación o para la adquisición de habilidades básicas de la programación de computadora es involucrar a los

estudiantes en actividades de diseño para una solución, es decir, aprendiendo al hacer y ser guiados paso a paso en este proceso, utilizando el aprendizaje basado en problemas, haciendo uso de lenguajes de programación o diagramas de flujo de datos (DFD), como herramientas de representación de la solución. Sin embargo, esto conlleva a que el estudiante tenga una comprensión de estructuras previamente desarrolladas, forzándolo a integrar en su lenguaje conceptos de un nivel más elevado, es decir se pretende que exista un aprendizaje, pero no un análisis de cómo se generan estas estructuras.

Este método ha permitido generar habilidades para la resolución de problemas en computación, pero no ha logrado desarrollar la habilidad de pensamiento analítico, tal es el caso que los estudiantes dedican más del 80% de su tiempo frente a la computadora, generando código y solo 20% al análisis y diseño de los algoritmos [5], confundiendo el proceso de programación con el de codificación.

Un método relativamente nuevo es el basado en el desarrollo del pensamiento analítico pero orientado a la computación [6], llamado pensamiento computacional. Sin embargo, al ser un concepto reciente, no se ha establecido una definición unificada [7, 8]. Para este trabajo se toma la definición de Google en su curso de "Pensamiento Computacional" en el que establece 4 formas de pensamiento que son usadas para la escritura de programas de cómputo [9]:

- Descomposición: Separar o dividir los datos, procesos o problemas en partes más pequeñas y manejables.
- Reconocimiento de patrones: Observación de patrones, tendencias y regularidades en los datos.
- Abstracción: Identificar los principios generales que permiten generar patrones.
- Diseño de Algoritmos: Desarrollo de las instrucciones paso a paso para resolver el problema identificado y otros problemas similares.

En la literatura se pueden encontrar distintas estrategias para la enseñanza del reconocimiento de patrones, como el uso de tarjetas [10], de igual forma el de abstracción y el diseño de algoritmos [9]. Sin embargo, la etapa de descomposición, aunque parece sencilla, requiere de trabajos que describan de forma sistematizada su abordaje, siendo la etapa más compleja para los estudiantes [11]. Además, es el concepto menos analizado en la literatura, con solo un 6% [12]. En el curso de pensamiento computación de Google [9], se menciona el concepto de descomposición, pero queda solo indicado.

En este trabajo para profundizar sobre la etapa de descomposición se retoma un elemento que es esencial en cualquier método de enseñanza, es decir el aprendizaje, descrito como un mecanismo personal que para ser observado requiere de una aproximación directa con procesos implicados como el pensamiento, sentimiento y acción, así como de habilidades que permiten su permanencia [13].

Algunos autores definen al pensamiento como una función psíquica a través de la cual un individuo usa representaciones, estrategias y operaciones frente a situaciones o eventos de orden real, ideal o imaginario, que se utiliza constructivamente en el mundo del individuo. Desde la orientación cognitivista que retoman los psicólogos se menciona que para describir el pensamiento es necesario observarlo desde la teoría del procesamiento de información, lo que implica la transformación de aquella información que proviene de diferentes canales sensoriales en información que puede ser procesada, es decir, pensar en un lenguaje que sea entendible, pensando en términos de algoritmos, y que posteriormente pueda retomarse para emplearse [14].

A partir de la revisión de la literatura se observa una amplia gama de artículos y publicaciones acerca de cómo podría desarrollarse el pensamiento en los estudiantes universitarios, pero cabe resaltar la existencia de tipos de pensamientos, y que uno de los necesarios para la formación académico-profesional de los estudiantes, es el pensamiento analítico.

El pensamiento analítico permite encuadrar la realidad, es decir, crear representaciones de la realidad a partir de variables que permite obtener datos. La capacidad del individuo para comprender una situación, organizar sus elementos de forma sistemática y determinar sus interrelaciones. De esta forma, se desarrolla independencia cognitiva, puesto que se realizan análisis que traspasan los conocimientos ya impuestos o establecidos.

Se han descrito ocho elementos que permiten desarrollar el pensamiento analítico [15], pero para que exista una lógica en su desarrollo; en primer lugar debe presentarse un propósito, es decir, meta, objetivo, lo que desea lograr; pregunta, expone el problema o asunto que guía; recopilación de información, los hechos, datos, evidencia o experiencias que se usan para descifrar las cosas; inferencias, interpretaciones o conclusiones; suposiciones, creencias que operan a nivel subconsciente o inconsciente, pero deben justificarse con evidencia sólida; conceptos, ideas, teorías, leyes, principios o hipótesis que se usan para tratar de hacer sentido de las cosas; punto de vista, perspectiva o enfoque.

Retomando la conceptualización del pensamiento analítico y sus características se identifica que existe una relación entre tipo de pensamiento y el proceso de

aprendizaje, de tal manera que es necesario una metodología de enseñanza que permita desarrollar las habilidades del pensamiento analítico mientras los estudiantes adquieren el conocimiento de programación.

Desarrollar el pensamiento analítico implica mecanismos poco complejos, de hecho, conlleva utilizar elementos y herramientas que ya se utilizan en las actividades diarias, un ejemplo claro es lo que se presenta a partir de pensamiento computacional, y que se planteó en este proceso metodológico con los estudiantes.

2. Propuesta Metodológica de Enseñanza de la programación de computadoras basada en el proceso de pensamiento computacional complementado con pensamiento analítico

En la academia es común identificar el proceso de resolución de problemas de pensamiento computacional [9], como el método tradicional de enseñanza. Es decir, que el proceso de desarrollo de un algoritmo en la industria o por un experto, es el mismo proceso que se retoma para la enseñanza o desarrollo de la habilidad. La hipótesis de trabajo planteada es que el proceso metodológico para la enseñanza de la programación propuesto debe brindar a los estudiantes desde el inicio una contextualización para la configuración de su propio lenguaje, lo que permite el desarrollo de pensamiento analítico para la construcción de soluciones cada vez más complejas utilizando herramientas clave como los DFD.

Por lo que se propone un orden diferente en las etapas de enseñanza, iniciando con la descomposición o análisis del problema, continuando con la identificación de las estructuras de control o proceso de validación estructural, posteriormente el reconocimiento de patrones, abstracción y diseño de algoritmos, finalizando con la codificación e implementación.

Para la implementación del proceso metodológico para la enseñanza de la programación propuesto, se seleccionó un grupo de 25 estudiantes de ingeniería en tecnologías interactivas y animación digital de primer semestre, que cursaron la asignatura de introducción a los sistemas computacionales durante el periodo agosto-diciembre 2016.

El proceso metodológico para la enseñanza de la programación propuesto se implementó desde el inicio de la asignatura antes mencionada, y se divide en 5 etapas, sub-divididas en diversas actividades, de las cuales se obtuvieron retroalimentaciones

para observar el desarrollo de pensamiento analítico y cada uno de sus ocho elementos. Cada uno de los ejercicios realizados por los estudiantes son acompañados de comentarios acerca de cuáles son las principales problemáticas durante el desarrollo de la actividad, qué piensan del ejercicio en sí, qué están aprendiendo durante dicha actividad, si están logrando resolver la actividad y cómo esto abona para el aprendizaje de la programación.

2.1. Primera Etapa de Aprendizaje por Problemas (Punto de Referencia)

La primera parte del curso comprende aproximadamente un mes de 3 horas a la semana. Se presenta a los alumnos la solución de problemas en computación con un enfoque de aprendizaje por problemas, utilizando el DFD como herramienta de representación y solución de problemas [16]. Esto permitió tener referencia sobre su conocimiento y habilidades previas respecto a la solución de problemas de cómputo.

A partir de lo anterior, se reflexionó acerca de ¿Cuáles son las habilidades que acompañan a los estudiantes en el proceso de aprendizaje para la comprensión de nuevos conocimientos, específicamente en el área de la programación? Una de las principales observaciones que se registraron durante el desarrollo de esta etapa del curso, fue que los estudiantes no podían diseñar algoritmos utilizando el DFD como herramienta de captura. Es decir, que a pesar de que algunos de ellos argumentaban tener conocimiento acerca de la programación no lograban completar con éxito las tareas que se implementaron en el aula.

2.2. Segunda Etapa de Enseñanza para el Aprendizaje: La Descomposición

El principal reto al que se enfrenta el estudiante es la etapa de descomposición del problema en elementos manejables o en problemas más pequeños, que resulta complejo para estudiantes principiantes, ya que no es claro hacia quién van dirigidas las instrucciones [17]. Por lo que se utilizaron elementos kinestésicos e interactivos para que los estudiantes comprendieran los conceptos con ideas que fueran tangibles para la creación de las propuestas de solución.

El grupo se dividió en equipos de 4 a 6 personas, y a cada persona del equipo se le asignó alguno de los siguientes roles: 1) ALU, 2) REGISTRO 3) USUARIO 4) DIRECTOR.

Se establece que la persona con el rol de ALU solo sabe hacer suma, resta, multiplicación y división (esto marcó el límite de capacidad de operaciones), y solo se pueden almacenar o proveer números (se establece el concepto de tipos de datos), y los registros son espacios, en los que se puede almacenar un valor entero. Por último, un DIRECTOR, que es la primera aproximación de lo que es un sistema operativo o el control interno de recuperación de instrucciones en una computadora. Entonces se solicitó al estudiante que resolviera un problema de sumatoria de 3 números, y quien tomó el rol de DIRECTOR, fue quien dio las instrucciones a la ALU, la solicitud de datos al usuario. Se solicitó también al DIRECTOR ir registrando cada instrucción brindada en una hoja de papel. Posterior al ejercicio se trabajó con todos los grupos para construir un estándar de instrucciones de manera que las instrucciones que diera el DIRECTOR de un grupo funcionarán para las de otro grupo. El grupo de estudiantes generó su propio lenguaje:

i1: USUARIO-DAME-VALOR

i2: REGISTRO "X" TE-DOY-VALOR-QUE-TRAIGO

i3: REGISTRO "X" DAME-VALOR

i4: ALU TE-DOY-EL-VALOR-QUE-TRAIGO

i5: ALU [/|+|-|*]

i6: ALU TE-DOY [INTEGER]

i7: ALU DAME-RESULTADO

Posteriormente se plantearon ejercicios con retos numéricos que fueron resueltos paso a paso usando estas instrucciones, intercambiando los roles, e intercambiando los DIRECTORES entre equipos.

2.3. Tercera Etapa de Enseñanza para el Aprendizaje: Las Estructuras y Validez Estructural

El DFD es una representación no simbólica con ventajas sobre las descripciones lingüísticas (lenguaje), por lo que en esta etapa se utiliza para identificar las estructuras de control propuestas por Dijkstra [18] en el paradigma de la programación estructurada. Se generan en cartas de tamaño de media cuartilla cada una de las estructuras de control (estructura selectiva, iterativa y secuencial), para fortalecer la comunicación de los conceptos bajo un enfoque de objetos físicos llamado manipulativos, tal como se hace en matemáticas para lograr que el estudiante se enganche en el aprendizaje de estas y que posteriormente en la cuarta etapa podrán ser utilizadas para la identificación de

reconocimiento de patrones del pensamiento computacional como alternativa a otras ya existentes en la literatura como multimedia, gráficos o gamificación [10, 19].

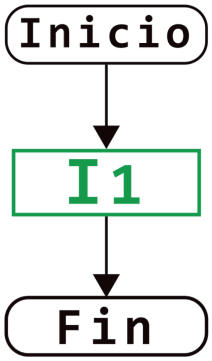


Figura 1: Estructura de control básica de Inicio-Proceso-Fin que describe el nivel más alto de abstracción de un algoritmo.

Se plantearon algoritmos sin contemplar el proceso de descomposición o de análisis del problema, el objetivo fue poder identificar las estructuras de control básicas, y poder aplicar el proceso de abstracción que obtenga la estructura básica de INICIO-PROCESO-FIN (Ver figura 1). De no lograrse dicha estructura se dice que el algoritmo no es válido estructuralmente y por lo tanto no puede implementarse en ninguna computadora. De lograr obtener la estructura INICIO-PROCESO-FIN, entonces se dice que el algoritmo es estructuralmente válido. En la figura 2 se puede observar un caso de algoritmo no es estructuralmente válido, normalmente para implementarlo se hace uso de instrucciones de brinco como el GO-TO, generando códigos no estructurados de difícil o muy costoso mantenimiento.

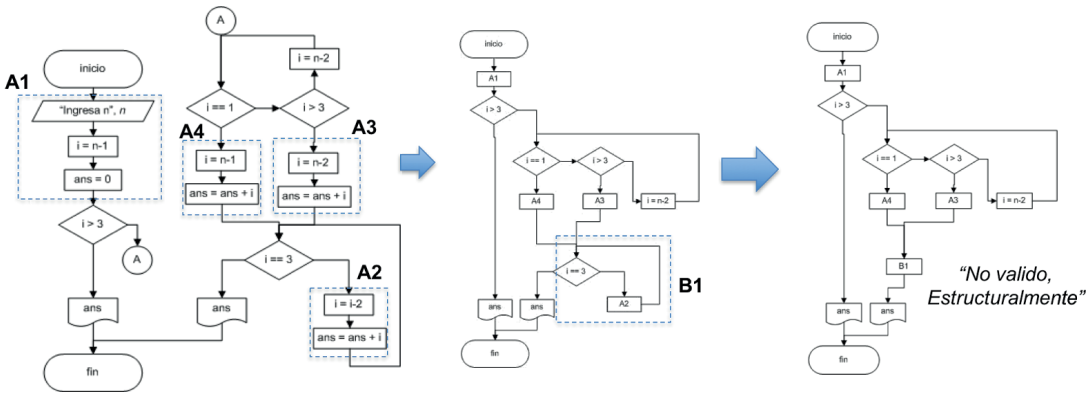
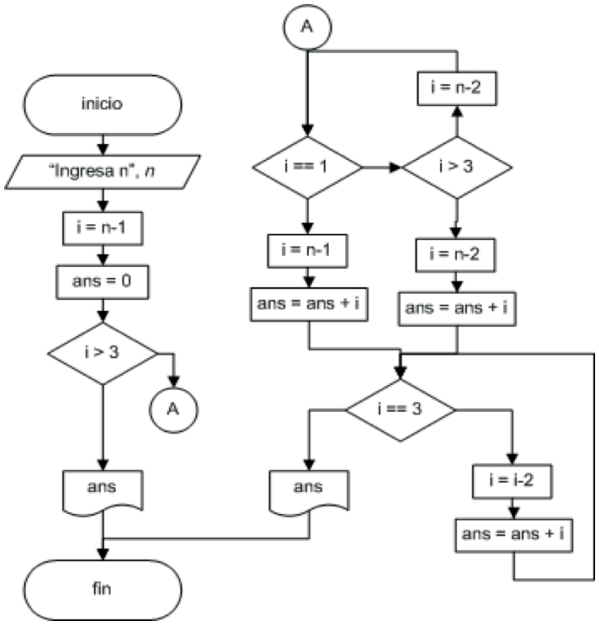


Figura 2: Proceso de validación estructural de un algoritmo.

2.4. Cuarta Etapa de Enseñanza para el Aprendizaje: El Reconocimiento de Patrones, Abstracción y Diseño de Algoritmo

En esta etapa se construye el diagrama de flujo para la solución propuesta utilizando los elementos estructurales presentados en la etapa anterior. Identificando los elementos que forman el algoritmo tomando en cuenta los elementos identificados de la etapa de descomposición.



i	ans	n
7	0	8
5	7	
3	12	
1	15	
7		
5		
3		

Figura 3: Proceso de validación lógica del algoritmo usando prueba de escritorio.

2.5. Quinta Etapa de Enseñanza para el Aprendizaje: La Codificación o Implementación

En la Quinta etapa se genera el código de computadora iniciando con la estructura INICIO-PROCESO-FIN, y reconstruyendo (en la imagen 4 de izquierda-derecha) agregando el código que corresponda para cada estructura básica, obteniendo en automático el código asociado al algoritmo diseñado.

3. Resultados

Para la interpretación de los datos obtenidos se retoma el análisis de contenido, el cual "se basa en la lectura (textual o visual) como instrumento de recogida de información,

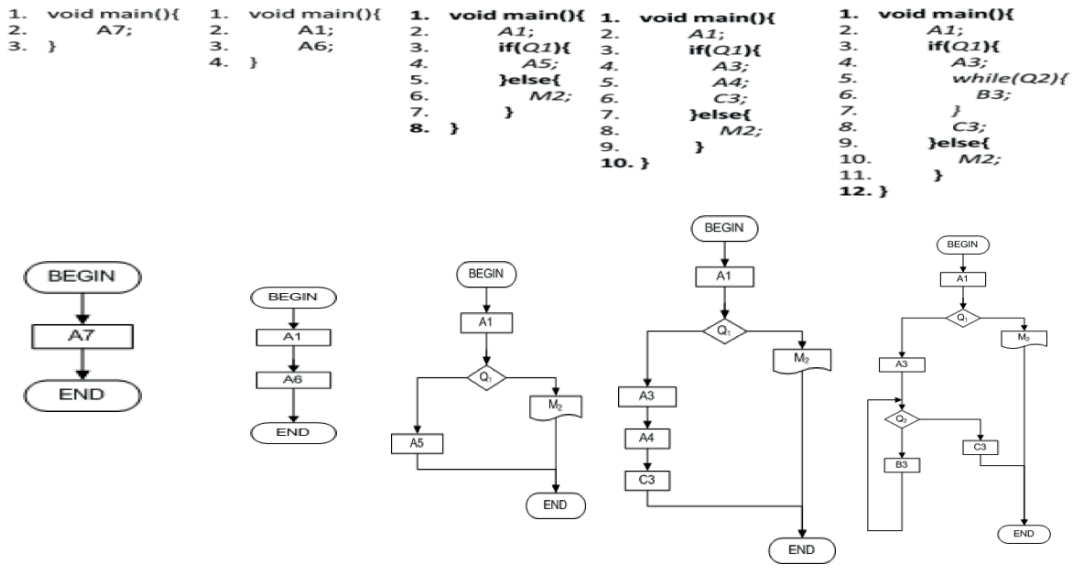


Figura 4: Proceso de codificación del algoritmo (Proceso de Abstracción inverso).

lectura que a diferencia de la lectura común debe realizarse siguiendo el método científico, es decir, debe ser, sistemática, objetiva, replicable, y válida" [20].

Los resultados obtenidos durante el curso fueron los siguientes:

La Tabla 1 permite identificar habilidades que los alumnos refieren haber desarrollado durante el primer parcial a partir del uso de metodología de aprendizaje tradicional, la primera columna (izquierda) refiere a cada uno de los alumnos, lo cual se utilizó por cuestiones de confidencialidad, la columna nombrada habilidades hace énfasis de las dos habilidades que se identificaron durante todo curso, a partir de evaluaciones y retroalimentaciones realizadas por escrito por parte de los alumnos se estableció la repuesta ¿¿SI/NO¿¿; y la tercer columna (derecha) describe comentarios redactados por parte de los estudiantes acerca de lo que consideran necesario para desarrollar las habilidades antes mencionadas.

Por su parte, la Tabla 2 se diseñó a partir de los componentes del pensamiento analítico, la cual permite observar la cantidad de alumnos que tenían problemas con el desarrollo de los componentes y su relación directa de la metodología de aprendizaje tradicional (utilizada durante el primer parcial) y la metodología propuesta (utilizada en el segundo y tercer parcial). La primera columna (izquierda) describe los ocho componentes del pensamiento analítico, la segunda columna (centro) describe la cantidad de alumnos que tenían problemas durante el primer parcial, y la tercer y cuarta columna nuevamente la cantidad de alumnos que tenían problemas para desarrollar las categorías.

Finalmente, la Tabla 3 describe una relación con los componentes del pensamiento analítico y el desarrollo del curso (contenido), esta relación está basada en los comentarios que realizaron los estudiantes a partir de las retroalimentaciones finales del curso (a partir de que se utilizó la metodología propuesta en este documento). La primera columna (izquierda) contiene los componentes del pensamiento analítico, y la segunda columna (derecha) contiene los comentarios de los estudiantes.

Tabla 1: Muestra de habilidades y retroalimentación de los alumnos al finalizar el primer parcial.

Alumno	Habilidades		Comentarios para mejorar el proceso de aprendizaje concluyendo el primer parcial del curso
	Análisis de problemas	Diseño de Algoritmos	
A	SI	NO	Es necesario que se explique paso por paso el proceso de desarrollo
B	NO	NO	Se necesita explicar paso por paso además el porqué de cada paso
C	NO	NO	Ver cómo se resuelve paso por paso un algoritmo para entenderlo mejor
D	SI	SI	Explicar paso por paso, leer, analizar y practicar
E	NO	NO	Desarrollar un mejor análisis y obtener el paso por paso del diseño de algoritmos
F	NO	NO	Explicar paso por paso y pensar en un mecanismo para esos pasos desarrollarlos más rápido
G	SI	SI	Explicar paso por paso y relacionarlos con acciones
H	NO	NO	Saber comprender los problemas, y expresar la solución

Tabla 2: Desarrollo de una nueva metodología y el desarrollo de los componentes del pensamiento analítico.

	Alumnos que refieren tener problemas para el desarrollo de:	Alumnos que refieren en sus comentarios tener problemas con el desarrollo de:	
Componentes del pensamiento analítico	Primer parcial con uso de metodología tradicional	Segundo parcial con uso de nueva metodología	Tercer parcial con uso de nueva metodología
identificar el propósito	20	6	4
comprender la pregunta	20	6	4
recopilar información	20	6	4
hacer inferencias	20	6	4
crear suposiciones	19	5	3
establecer conceptos	19	5	3
identificar las consecuencias	11	5	3
generar un punto de vista	11	5	3

Tabla 3: Comentarios finales de los estudiantes en relación con el desarrollo del curso y su principal aporte a los componentes del pensamiento analítico.

Componentes	Contenido del curso: comentarios finales de los estudiantes
propósito	forma de trabajar clara, se explica punto por punto el proceso
pregunta	resolución de dudas antes de resolver el problema
recopilar información	uso de técnicas como fórmulas, ejemplos numéricos, conocimientos de los compañeros sobre ejemplos reales
inferencias	aprendizaje, identificación de las áreas de oportunidad
suposiciones	es necesaria la práctica constante, la lógica aplicada permite resolver problemas
conceptos	formular bien el problema permite su resolución,
consecuencias	desarrollo de habilidades, mejora de habilidades ya presentes,
punto de vista	la retroalimentación es necesaria para aprender, la perspectiva de otro puede ayudar bastante para entender los problemas y su resolución

4. Discusión

El curso con estudiantes de ingeniería tenía como finalidad que los estudiantes obtuvieran los conocimientos necesarios para la programación, pero durante el transcurso del primer parcial se observaron una serie de problemáticas que una vez analizadas conllevaron a la implementación de estrategias distintas, las cuales brindaron resultados positivos como se muestra en la Tabla 1, pero que además, desarrollaron y potencializaron en la mayor parte de los estudiantes habilidades como el análisis de problemas y la comprensión para el diseño de algoritmos.

Ahora bien, durante el desarrollo del curso los estudiantes describen que existen menos problemáticas en su proceso de aprendizaje como se muestra en la Tabla 2 ya que al implementarse la propuesta metodológica de enseñanza los estudiantes desarrollaron y/o potencializaron habilidades que les permiten resolver los problemas planteados, que además dichos problemas/ejercicios presentaban un grado mayor de complejidad. y se identificó una relación de los componentes del pensamiento analítico y el uso de la nueva metodología de enseñanza, se retomaron comentarios redactados por los estudiantes, se analizó el contenido y se identificaron palabras que describen cada uno de los componentes, así como las características que ellos observaban y que les ayudaban para resolver los ejercicios, ejemplo: "me costó mucho comprender el problema, no entendía por dónde empezar, cómo podría resolverlo, pero una vez que pude entender y hacer un análisis el diseño para resolverlo fue más sencillo, aunque la estructura aún se me complica mucho porque no entiendo muy bien cómo plantearlas", de tal manera que el alumno en sus comentarios y de acuerdo a los revisado de los ocho componentes del pensamiento analítico, se identifican problemas

en la pregunta, ya que el estudiante no identifica en ese momento los pasos adecuados para implementar la solución al problema general; y conceptos, porque el estudiante al no comprender el problema no logra identificar a qué componente debe recurrir y retomar para solucionar el problema en tiempo y forma. Al final del curso se les solicitó nuevamente a los estudiantes redactar comentarios, de todos los ejercicios y del curso en general, la Tabla 3 permite visualizar cuáles fueron las palabras que se repiten por parte de los estudiantes.

Para la implementación de la metodología propuesta y cada una de las cinco etapas, se establece lo siguiente: Etapa 1, (punto de referencia), en esta etapa se aplicó la metodología de pensamiento computacional que comúnmente se usa en la academia, por lo que solo es usada como referencia para este caso de estudio y poder identificar los niveles de desarrollo de la capacidad de abstracción de los estudiantes, por lo que no se requiere aplicar en nuevos cursos; Etapa 2, (descomposición), ésta etapa es el elemento que permite al alumno contextualizarse con su propio lenguaje, por lo que s importante no utilizar ningún lenguaje de programación existe, esto ocurrirá en la etapa 5; Etapa 3, (estructura y validez), en esta etapa es importante recalcar el uso de elementos visuales que permitan al estudiante construir elementos abstractos a partir de elementos concretos; Etapa 4, (diseño de algoritmos), en esta etapa debe dedicarse el tiempo suficiente a la construcción de algoritmos utilizando las estructuras aprendidas en la etapa 3; Etapa 5, (codificación), esta etapa puede inclusive dejarse para otro curso, agregando más tiempo a las etapa 2,3 y 4, ya que esta etapa se refiere al aprendizaje de un lenguaje en particular..

La metodología propuesta para la enseñanza de la programación se desarrolló a partir de un proceso empírico, por lo cual para su continuación es necesario: aumentar la muestra de participantes, validar la propuesta metodológica, diseñar instrumentos de recolección de información estadística, para la evaluación antes y después de la implementación en un nuevo curso.

Referencias

[1] Franklin D. Putting the Computer Science in Computing Education Research. Commun ACM. New York, NY, USA: ACM; 2015;58: 34–36.

[2] Bocconi S, Chioccariello A, Dettori G, Ferrari A, Engelhardt K, Kampylis P, et al. EXPLORING THE FIELD OF COMPUTATIONAL THINKING AS A 21ST CENTURY SKILL. EDULEARN16 Proceedings. IATED; 2016. pp. 4725–4733.

[3] P EEJ, Wong B, Berry M. The Roehampton Annual Computing Education Report. 2015 data from England [Internet]. University of Roehampton; 2016 Dec. Available: https://www.researchgate.net/profile/Billy_Wong/publication/311595274_The_Roehampton_Annual_Computing_Education_Report_2015_data_from_England/links/5850296608ae4bc8993b6845.pdf.

[4] Pappas IO, Giannakos MN, Jaccheri L. Investigating Factors Influencing Students' Intention to Dropout Computer Science Studies. Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '16. 2016. doi:10.1145/2899415.2899455

[5] Dolado AR, Arana LS. Metodología de programación. Principios y aplicaciones. Editorial Club Universitario; 2004.

[6] Wing JM. Computational thinking. Commun ACM. 2006;49: 33–35.

[7] Weintrop D, Beheshti E, Horn M, Orton K, Jona K, Trouille L, et al. Defining Computational Thinking for Mathematics and Science Classrooms. J Sci Educ Technol. Springer Netherlands; 2016;25: 127–147.

[8] Barr V. Bringing Computational Thinking to K-12: What is involved and what is thye role of the computer science education community? ACM Inroads. 2011;2: 48–54.

[9] Computational Thinking for Educators - Course [Internet]. [cited 28 Jul 2017]. Available: https://computationalthinkingcourse.withgoogle.com/

[10] Aggarwal A, Gardner-McCune C, Touretzky DS. Evaluating the Effect of Using Physical Manipulatives to Foster Computational Thinking in Elementary School. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. ACM; 2017. pp. 9–14.

[11] Ho C-H. Some phenomena of problem decomposition strategy for design thinking: differences between novices and experts. Design Studies. 2001;22: 27–45.

[12] Kalelioglu F, Gülbahar Y, Kukul V. A framework for computational thinking based on a systematic research review. Baltic Journal of Modern Computing. University of Latvia; 2016;4: 583.

[13] del Carmen González Torres FTM&. M. Aprendizaje autorregulado: presente y futuro de la investigación. Revista Electrónica de Investigación Psicoeducativa. 2004;2: 1–34.

[14] Melgar SA. El pensamiento: Una definición interconductual. Revista de Investigación en Psicología. Julio, 2000;3: 24–38.

[15] Nosich GM. Aprender a pensar: pensamiento analítico para estudiantes. Pearson Educación; 2003.

[16] Cairo O. Metodología de la programación. ALFAOMEGA GRUPO EDITOR S.A. DE C.V., editor. 2003.

[17] Yadav A, Stephenson C, Hong H. Computational Thinking for Teacher Education. Communications of the ACM. april,2017;60: 55–62.

[18] Buhr PA. Understanding Control Flow: Concurrent Programming Using μC++. Springer; 2016.

[19] Komar M. Teaching Math with Everyday Manipulatives: Grades 4-6. On The Mark Press; 2007.

[20] Neuendorf KA. The Content Analysis Guidebook. Sage, editor. 2016.