Pedagogía del Pensamiento Computacional desde la Psicología: un Pensamiento para Resolver Problemas

Pedagogy of Computational Thinking from the Psychology: a Problem-Solving Thinking

Recibido: 28/07/2020 | Revisado: 28/09/2020 | Aceptado: 30/09/2020 | Publicado: 22/12/2020

Beatriz Ortega-Ruipérez
Universidad Internacional de La Rioja (España)
beatriz.ortega.ruiperez@unir.net
http://orcid.org/0000-0002-3822-5745

Resumen: El pensamiento computacional debe entenderse como un pensamiento estratégico para resolver problemas, más allá de su vinculación con la programación. Por ello, es necesario abordar la estructura de este pensamiento, a través de sus procesos cognitivos, para obtener una definición operativa que permita a la pedagogía de este pensamiento abordarlo adecuadamente en las aulas, independiente del recurso que se utilice para su desarrollo. En este artículo se determinan cinco procesos inherentes a este pensamiento, identificando únicamente los que siempre se ven empleados, y se definen de forma operativa desde una perspectiva psicológica y también pedagógica. Estos procesos son abstracción, generalización, evaluación, creación de algoritmos y descomposición del problema. De estos procesos se puede comprobar una relación inherente de los cuatro primeros a la resolución de problemas, mientras que la descomposición se puede considerar no fundamental en la resolución de problemas y, por tanto, propia del pensamiento computacional. Gracias a este análisis y categorización, se comprueba que la descomposición es el proceso clave y central de este pensamiento, con lo que la didáctica del pensamiento computacional debe abordarse siempre desde la descomposición de problemas o tareas que permitan la simplificación del resto de procesos implicados, con diferentes recursos como la programación, el juego, la resolución de problemas, o la creación de proyectos.

Palabras clave: Pensamiento, resolución de problemas, cognición, pensamiento computacional, lenguaje de programación, estrategias educativas.

Abstract: Computational thinking should be understood as problem-solving thinking, beyond its link to programming. Therefore, it is necessary to address this thought's structure through its cognitive processes to obtain an operational definition that allows this thought's pedagogy to be adequately addressed in the classroom, regardless its development source. In this article, five processes inherent to this thought are determined, identifying only those that are always used. These processes are operatively defined from a psychological and a pedagogical perspective. The processes are abstraction, generalization, evaluation, creation of algorithms and decomposition of the problem. The first four processes present an inherent relation to problem-solving that can be verified, while, the last one, decomposition, can be considered not fundamental in problem solving and, therefore, linked to computational thinking. The performed analysis and categorization prove that decomposition is the key and central process of computational thinking. Therefore, computational thinking teaching must always be approached from the decomposition of problems or tasks that allow the simplification of the rest of the processes involved. This should involve different resources such as programming, playing games, problem-solving, or the creation of projects.

Keywords: Thinking, problem solving, cognition, computational thinking, programming languages, educational strategies.

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10

Introducción

En los últimos años se ha visto la introducción de la programación como materia en la enseñanza obligatoria, o su reclamo como algo necesario, como son los casos en Europa de Reino Unido (Csizmadia *et al.*, 2015), Alemania (Delcker y Ifenthaler, 2017), Francia (Académie des Sciences, 2013), Noruega (Bocconi, Chioccariello y Earp, 2018), o en España, incluida de forma obligatoria en la Comunidad de Madrid (Valverde-Berrocoso, Fernández-Sánchez y Garrido-Arroyo, 2015) como proyecto piloto, entre otros.

Este planteamiento de considerar la programación como algo necesario para los alumnos en el siglo XXI, ha acelerado la aceptación del pensamiento computacional (PC) como un concepto imprescindible en la enseñanza, sin dar tiempo suficiente a su investigación: qué procesos implica este pensamiento, para qué puede ser realmente útil y cómo debemos abordarlo en educación, más allá de enseñar a programar.

Además, como se lleva instando estos últimos años, enseñar a programar no implica desarrollar el PC, ya que, de acuerdo con Zapata-Ros (2015), se debe tener en cuenta el enfoque que se adopta: si se trata de un enfoque reproductivo o un enfoque productivo.

Por otra parte, no se debe olvidar que este pensamiento se considera como un conjunto de procesos útiles en la resolución de problemas complejos (Barr y Stephenson, 2011; Grover y Pea, 2013; Voogt, Fisser, Good, Mishra y Yadav, 2015), y que por tanto se debe ver como una estrategia o heurístico que se puede aplicar en el proceso de resolución de problemas (Angeli *et al.*, 2016), pero que además se podría utilizar para simplificar la forma que abordamos la representación de este (Kwisthout, 2012; Wing, 2014).

Por ello, como pensamiento útil para la resolución de problemas, más allá del contexto informático, se debe tener en cuenta la posibilidad de desarrollar este pensamiento con otros medios, como es el enfoque *unplugged*, o por medio de proyectos de edición de vídeo, creación de artefactos digitales, etc. (Kong y Abelson, 2019).

El objetivo de este artículo es proporcionar un modelo sobre el pensamiento computacional y unas directrices pedagógicas para asegurar el desarrollo del PC a través de actividades de diversa índole que trabajen los procesos implicados.

Para ello, en primer lugar, se realiza un análisis desde la psicología diferenciando entre procesos de pensamiento para emplear este pensamiento, y habilidades o competencias. Y así proponer un modelo teórico del pensamiento computacional comprendido desde su relación con la resolución de problemas, pudiendo crear además una definición operativa, algo que todavía no se ha consensuado en torno a este concepto (Bocconi *et al.*, 2016).

Por último, se proponen unas directrices pedagógicas para asegurar el trabajo de todos los procesos operativizados en el pensamiento computacional, tanto desde la perspectiva del aprendizaje de la programación, como desde su trabajo desde otras técnicas educativas como la resolución de problemas o la creación de proyectos.

Descomponiendo el pensamiento computacional

Para delimitar este pensamiento, se han seleccionado los cinco marcos más actuales y relevantes (Tabla 1), marcos que se basan en la revisión de marcos anteriores. Se considera que al seleccionar estos marcos actuales, basados en la revisión de otras propuestas previas, no es necesario analizar más allá de estas cinco, ya que el resto se basan en la revisión de los mismos autores, como la propuesta original y ampliada de Wing (2006, 2014, 2008, 2011), la de Barr y Stephenson (2011), o la de Brennan y Resnick (2012), por citar algunas.

El marco conceptual de Selby y Woollard (2013) es la primera propuesta que intenta operativizar los procesos cognitivos incluidos en la definición del PC para conseguir unificar el término, de forma que resulte más fácil abordarlo en la enseñanza. Para ello, recogen algunas propuestas relevantes de años

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10

anteriores a su publicación, y analizan los términos más frecuentes en las descripciones. En el caso del marco conceptual de Csizmadia *et al.* (2015), se basan en las definiciones propuestas por Wing (2011), la autora que acuñó el concepto.

Por otra parte, la propia autora realiza una actualización reciente (Wing, 2017). Se comprueba que en esta actualización no se ha incluido ningún proceso posterior. Aunque destaca la abstracción como clave del PC.

La Sociedad Internacional de Tecnología en Educación (ISTE) por su parte, propuso junto a la Asociación de Maestros de Ciencias de la Computacional de Estados Unidos (CSTA) una definición operativa para conocer los procesos implicados en el pensamiento computacional (ISTE y CSTA, 2011), que ha sido trabajado desde entonces, con lo que se ha logrado una definición posterior (ISTE, 2016), definición que incluye todos los procesos que dicen estar implicados en 2011, a excepción de la simulación y el paralelismo.

Bocconi *et al.* (2016) revisan varias de las propuestas previas más relevantes, como las Barr y Stephenson (2011), Grover y Pea (2013) y Selby y Woollard (2013), entre otras. Estas propuestas han sido utilizadas en muchos de los estudios y experiencias que se han realizado sobre el pensamiento computacional, y que gracias a Bocconi *et al.* (2016) han sido integradas.

Por su parte, Shute, Sun, y Asbell-Clarke (2017) crean su propia definición partiendo de dos propuestas muy relevantes, como son la propuesta de Brennan y Resnick (2012), quienes diferenciaban entre conceptos, prácticas y perspectivas del pensamiento computacional, y la propuesta de Weintrop *et al.* (2016).

Desde 2013 se ha intentado consensuar los procesos de este pensamiento a través de revisiones de la literatura. Sin embargo, en la actualidad cada autor que investiga sobre pensamiento computacional elige la propuesta que más le encaja, asumiendo que existe un vacío entorno a la definición operativa de este pensamiento.

Tabla 1

Procesos del pensamiento computacional según las revisiones teóricas más relevantes sobre el concepto

Autoría	Habilidades del pensamiento computacional
Selby y Woollard (2013)	Abstracción, descomposición, algoritmización, evaluación, generalización.
Csizmadia *et al*. (2015)	Algoritmización, descomposición, generalización (patrones), abstracción (representaciones), evaluación.
ISTE (2016)	Abstracción, algoritmización, recogida de datos, análisis de datos, representación de datos, descomposición de problemas, automatización.
Bocconi *et al.* (2016)	Abstracción, algoritmización, automatización, descomposición, depuración, generalización.
Shute, Sun y Asbell-Clarke (2017)	Descomposición, abstracción, diseño de algoritmos, depurar, iteración, generalización.

Fuente. Elaboración propia.

Procesos inherentes al pensamiento computacional

El pensamiento, según Carretero y Asensio (2004), es aquello que contiene un conjunto de operaciones intelectuales como el razonamiento, abstraer, hacer generalizaciones, con el objetivo de la representación de la realidad, toma de decisiones o resolución de problemas. Siguiendo esta definición, y teniendo en cuenta los procesos superiores de pensamiento (Smith y Kosslyn, 2008), se deben diferenciar de forma clara los procesos de pensamiento incluidos en las propuestas sobre el pensamiento computacional y otros procesos y habilidades los cuales, aunque pueden formar parte de este pensa-

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10
Cuestiones Pedagógicas, *2*(29), 2020, pp.130-144

miento y se pueden trabajar de forma simultánea, no podrían ser considerados componentes básicos del PC. Por ello, se deben descartar los componentes que no puedan ser catalogados como procesos de pensamiento (Tabla 2).

Tabla 2

Análisis de procesos para seleccionar procesos de pensamiento y evaluar su solapamiento

Procesos	Aparición	Proceso superior	Solapamiento con otros procesos superiores
Abstracción	100% (5/5)	SÍ	Recogida y análisis datos
Algoritmización	100% (5/5)	SÍ	
Descomposición	100% (5/5)	NO	
Generalización	80% (4/5)	SÍ	Representación datos
Evaluación	40% (2/5)	SÍ	Recogida, análisis, representación datos
Automatización	40% (2/5)	NO	
Depurar	40% (2/5)	NO	
Iteración	20% (1/5)	NO	
Recogida de datos	20% (1/5)	SÍ	Abstracción y evaluación
Análisis de datos	20% (1/5)	SÍ	Abstracción y evaluación
Representación de datos	20% (1/5)	SÍ	Evaluación y Generalización

Fuente. Elaboración propia.

Como se puede apreciar, se encuentran cuatro de los procesos que no se pueden considerar procesos superiores de pensamiento: descomposición, automatización, depurar e iteración. En cuanto a la descomposición, aparece en todas las propuestas analizadas, mostrando su importancia como estrategia dentro del pensamiento computacional.

Según Ortega-Ruipérez (2018), la descomposición resulta ser un proceso fundamental del pensamiento computacional, ya que la clave de este pensamiento reside en descomponer un gran problema en pequeñas partes computacionalmente abordables, incluyendo cómo se aborda y representa el problema. Por tanto, la descomposición es una estrategia para facilitar los procesos superiores de pensamiento empleados en la resolución de problemas.

Respecto a la automatización, parece una destreza que se puede incluir al emplear este pensamiento, al estar además relacionado con las Ciencias de la Computación, pero no se puede concluir que sea algo necesario e inherente al pensamiento computacional, ya que no aparece de forma general en las propuestas, sino que más bien se trataría de una forma opcional de facilitar la resolución de problemas con este pensamiento.

Por su parte, la depuración y la iteración serían más bien prácticas muy relacionadas con la programación informática, prácticas que requieren de los procesos de pensamiento ya incluidos, ya sea la abstracción, evaluación y generalización; o los procesos relativos al tratamiento de datos.

Con este análisis, se puede comprobar que de los cuatro procesos excluidos, sólo uno de ellos se puede considerar fundamental en el pensamiento computacional: la descomposición.

Por la otra parte, sólo se consideran los procesos superiores de pensamiento, es decir, abstracción, evaluación, generalización, recopilación de datos, análisis de datos, representación de datos y creación de algoritmos.

Sin embargo, se ve cómo existe un solapamiento entre los procesos superiores de pensamiento propuestos, ya que por una parte tenemos abstracción, generalización y evaluación; y por otra recogida, análisis y representación de datos. Los primeros procesos aparecerían de forma combinada en los procesos relativos al tratamiento de datos (recogida, análisis y representación).

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10

Adicionalmente, existen otros intentos de definir el pensamiento computacional desde las habilidades cognitivas que incluye, como la que proponen Román-González, Pérez-González y Jiménez-Fernández (2017). Sin embargo, en esta propuesta se mezclan procesos superiores (razonamiento) con procesos básicos (procesamiento visual y memoria a corto plazo). En este caso, se entiende que se refiere al razonamiento lógico, pues existen muchos tipos como el informal, analógico, proposicional, etc. (Carretero y Asensio, 2008).

El pensamiento computacional en la resolución de problemas

Bassok y Novick (2012) definen la representación de un problema como el modelo construido con los componentes del problema que la persona ha captado o entendido. Incluyen el estado inicial del problema y el estado objetivo. Esto permite elaborar el proceso para la resolución, lo que implica la creación de un algoritmo que avance por los estados del problema hasta su estado objetivo o solución.

El pensamiento computacional facilita estos procesos al descomponer ese gran problema en problemas sencillos. Sin embargo, hay que tener en cuenta que resolver varios problemas simples no equivalen a la resolución de problemas complejos, tal y como sostienen Funke, Fischer y Holt (2017), y que el PC pretende descomponer un problema complejo, pero siempre con una interrelación significativa entre todos los pequeños problemas.

Como recogen Dörner y Funke (2017), la resolución de problemas complejos es un proceso dinámico con diferentes fases dentro de la adquisición del conocimiento y la aplicación de éste, ya que requiere una colección de procesos psicológicos superiores y autorregulados, que se emplean de forma recursiva entre la adquisición del conocimiento, que corresponde a la representación del problema, y la aplicación del conocimiento, correspondiente al proceso de resolución del problema (Herde, Wüstenberg y Greiff, 2016).

Esta interacción entre la adquisición y la aplicación del conocimiento sobre el problema, se puede considerar, según Fischer, Greiff y Funke (2017) similar a la idea de tratamiento de información, concretamente a la generación de información y la reducción de información, ya que una aplicación efectiva del conocimiento requiere una adquisición adecuada, es decir, una reducción y generación de información eficaz.

Se opta por los procesos de abstracción, evaluación y generalización, al haber conocido su relación con el tratamiento de datos, confirmando que el pensamiento computacional está compuesto de los procesos superiores de pensamiento: abstracción, evaluación, generalización y algoritmización, que se emplean en la solución de problemas independientemente de utilizar una estrategia computacional, y la descomposición, que resulta una estrategia clave para facilitar cómo se aborda un problema complejo, coincidiendo con lo que proponen otros autores como Looi, How, Longkai, Seow y Liu (2018).

Modelo teórico del pensamiento computacional

Se parte de la consideración de que la descomposición del problema es el aspecto clave que proporciona emplear este pensamiento, lo que conlleva que se aborde el problema de forma más sencilla, y por lo que tanto, se simplifique el esfuerzo cognitivo que supone trabajar con los datos de forma simultánea. Se entiende por tanto, que al tratar los datos de forma computacional, divididos en cómputos con significado independiente, se simplifica el esfuerzo cognitivo asociado a cada uno de estos procesos.

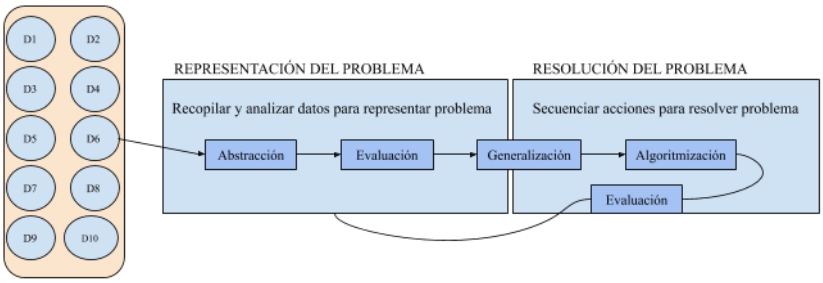
Por tanto, en la Figura 1 se han representado los procesos siempre presentes el proceso de resolución de problemas, aunque no se aplique el PC, suponiendo para la ejemplificación de la representación, un problema con 10 datos (D1-D10).

En la Figura 2 se ve cómo sería este proceso incluyendo la pieza clave: la descomposición, observando así la diferencia entre resolver el problema sin aplicar o aplicando el PC.

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10
Cuestiones Pedagógicas, *2*(29), 2020, pp.130-144

Figura 1

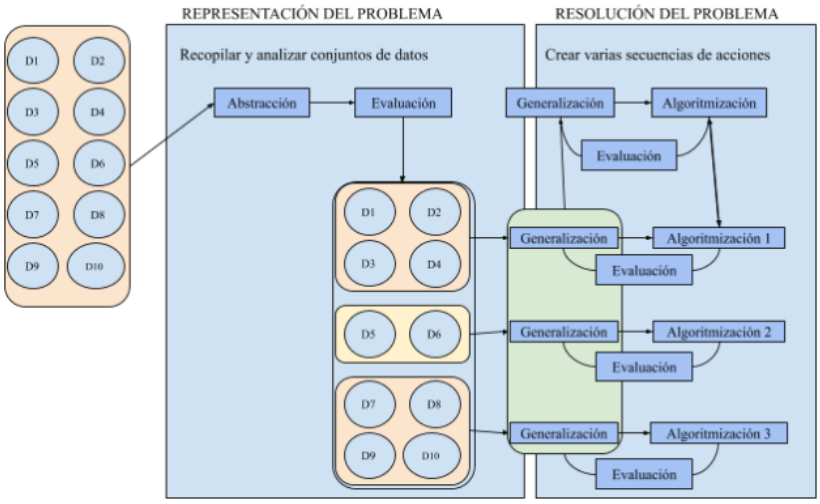
Representación simplificada de los procesos superiores de pensamiento en la resolución de problemas sin aplicar el pensamiento computacional (descomposición)



Fuente. Elaboración propia.

Figura 2

Representación simplificada de los procesos superiores de pensamiento en la resolución de problemas aplicando el pensamiento computacional



Fuente. Elaboración propia.

En el caso en el que se emplea la descomposición, la abstracción permite abordar diferentes conjuntos de datos con un significado conjunto. Esto además facilita la evaluación de los conjuntos de datos, frente a la evaluación de todos los datos como un gran conjunto.

Al evaluar cada conjunto, se generalizan pequeñas representaciones, a la vez de crear una representación conjunta para asegurar que el objetivo se mantiene, como se aprecia en la flecha que asciende del recuadro verde.

Con la representación del problema se puede generar un algoritmo general de resolución, pero lo verdaderamente útil de crear estas representaciones, es que se pueden crear varios algoritmos, y cada algoritmo centrarse en resolver cada parte.

La puesta en práctica de cada uno de los algoritmos puede evaluarse de forma computacional para, en el caso de que falle uno de los algoritmos, sólo se revisa esa parte del problema, sin afectar al resto. Además debe darse una evaluación general de cómo está avanzando el problema por los diferentes estados, por si hiciese falta una revisión general.

Definición operativa del pensamiento computacional

Como definición operativa del pensamiento computacional se obtiene que es una estrategia basada en la descomposición, que facilita la resolución de problemas complejos, y en particular los

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10

procesos de abstracción, evaluación, generalización y pensamiento algorítmico. En la Tabla 3 están las definiciones de cada proceso para su aplicación en este pensamiento.

Tabla 3

Definición de los procesos definidos operativamente en el pensamiento computacional

Habilidad	Definición
Descomposición	Permite dividir un gran problema o tarea en partes más simples e independientes que se pueden abordar individualmente.
Abstracción	Permite simplificar un concepto, identificando una idea a través del análisis de datos para desprender la idea general de los detalles.
Generalización	Permite la representación de un concepto o problema al establecer conexiones entre datos e identificar patrones comunes.
Evaluación	Permite analizar datos e ideas, a través de comparaciones lógicas que ayudan a verificar las ideas y acciones elaboradas.
Pensamiento Algorítmico	Permite la planificación de una secuencia lógica y limitada de acciones que conducen al logro de una tarea determinada.

Fuente. Elaboración propia.

Por otra parte, en la Tabla 4 se proponen cinco niveles de adquisición para cada uno de estos procesos. Con la creación de esta rúbrica se pretende facilitar la forma de abordar el pensamiento computacional en el aula, al permitir diseñar actividades que incluyan acciones para trabajar cada uno de los procesos implicados en el nivel que se quiera trabajar.

Tabla 4

Niveles de adquisición de los procesos del pensamiento computacional

Habilidad	1	2	3	4
Descomposición	Detecta dos grandes ideas en un mismo problema o tarea	Detecta las distintas partes e identifica un dato de cada idea	Detecta las distintas partes e identifica casi todos los datos	Detecta las partes de una tarea y los datos relativos a cada una
Abstracción	Extrae las ideas a través de un proceso guiado que ayuda a desechar detalles	Es capaz de extraer varias ideas por si mismo, desechando datos superficiales	Extrae casi todas las ideas, eliminando los detalles menos importantes	Extrae todas las ideas esenciales desprendiendo todos los detalles
Generalización	Representa ideas si ve claramente las conexiones entre los datos de cada una	Representa la idea general de la tarea o problema con fallos no determinantes	Crea representación del problema o tarea conectando la mayoría de ideas	La representación creada conecta de forma óptima todas las ideas extraídas
Evaluación	Es capaz de analizar datos sin relacionar unos con otros	El análisis de datos permite relacionar unos con otros	Analiza la mayoría de relaciones entre los datos relevantes	Analiza todas las relaciones entre los datos más relevantes
Pensamiento Algorítmico	Crea un algoritmo general e identifica cada subalgoritmo al alcanzar esa fase	Crea un algoritmo general y divide el trabajo en varias fases u objetivos	Puede crear los subalgoritmos de cada parte y evaluar si encajan entre sí	Crea un algoritmo general que guía los subalgoritmos de cada parte/subtarea

Fuente. Elaboración propia.

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10
Cuestiones Pedagógicas, *2*(29), 2020, pp.130-144

Directrices pedagógicas para desarrollar el PC

Relación entre el pensamiento computacional y la programación

Programar implica muchos de los procesos superiores que componen el pensamiento computacional, por lo que programación resulta un recurso ideal para desarrollarlo, si se trabaja de una manera productiva y no reproductiva (Zapata-Ros, 2015). Como afirma Bers (2017), la programación no es lo mismo que el pensamiento computacional, pero la programación recoge los objetivos y etapas que se emplean en el PC.

Para empezar, la programación debe abordarse desde una necesidad a resolver, desde la necesidad de que un programa automatizado nos facilite la realización de una tarea. Duncan, Magnuson y Votruba-Drzal (2014) denominan este tipo de programación como codificación, y proponen que debe basarse en plantear soluciones mediante un lenguaje de programación. Por tanto, la relación se da entre la codificación y el pensamiento computacional; y no con programación (González-González, 2018).

Así, el primer paso para la codificación es el diseño de un programa informático, el cual requiere un estudio de las diferentes funcionalidades que debe cumplir, y este aspecto es clave para asegurar que se trabaja desde una perspectiva computacional, ya que el diseño requiere una descomposición del gran programa en todas estas funcionalidades fundamentales para que funcione.

La programación consiste en la creación de un algoritmo, estableciendo todos los pasos necesarios para que el programa cumpla con sus objetivos, y que se puede representar a través de un diagrama de flujo que indique la relación y secuencia de las instrucciones incluidas.

Por otra parte, el programa diseñado ha de contar con elementos básicos que se permiten automatizar las instrucciones, como el uso de variables para guardar datos, el uso de funciones para guardar pequeños algoritmos recurrentes, o el uso de instrucciones de control, tanto condicionales como bucles.

Además, el desarrollo de un programa requiere de una constante monitorización que permite detectar errores y solventarlos, práctica conocida como depuración.

En la Tabla 5 se propone la relación en los procedimientos habituales en programación y los procesos del pensamiento computacional con mayor implicación en estos procedimientos.

Esto no quiere decir que los procesos del pensamiento computacional que se relacionan con cada procedimiento o práctica de programación se empleen únicamente en dicha práctica, sino que en esa práctica lo primordial es emplear el proceso superior de pensamiento relacionado.

Tabla 5

Procedimientos habituales en programación y su relación con el pensamiento computacional

Programación	Pensamiento computacional
Estudio de funcionalidades	Descomposición
Creación de un algoritmo	Pensamiento algorítmico
Diagrama de flujo	Abstracción, generalización
Variables	Abstracción
Funciones	Generalización
Condicionales	Evaluación
Bucles	Evaluación
Depuración	Evaluación, abstracción, generalización

Fuente. Elaboración propia.

Más allá de la programación, el pensamiento computacional se relaciona habitualmente con habilidades digitales, ya que se ha considerado un pensamiento fundamental para abordar cuestiones complejas relacionadas con la digitalización. Diversas organizaciones lo incluyen en la revisión de habilidades que proponen como necesarias en el siglo XXI, especialmente en el contexto digital.

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10

La Unión Europea propone la Competencia Digital, una competencia básica en la que se incluye el aprendizaje de la programación, relacionándolo con el pensamiento computacional como una habilidad para la expresión digital (Vuorikari, Punie, Carretero y Van Den Brande, 2016).

Por otra parte, la UNESCO incluyó el pensamiento computacional como parte de la Alfabetización Informacional y Mediática, de forma que las personas participen de forma activa y reflexiva de la cultura y la información digital (Gretter y Yadav, 2016).

En este sentido, autores como Adell, Llopis, Esteve y Valdeolivas (2019) muestran cómo la competencia digital puede desarrollarse al trabajar el pensamiento computacional a través de la tecnología.

Sin embargo, hay que tener en cuenta que trabajar programación, simplemente con tecnología, no garantiza su desarrollo. Para ello, se deben identificar qué procesos requieren las actividades, fundamentalmente la descomposición.

Desarrollo del PC a través de la codificación

La programación puede abordarse desde dos aproximaciones didácticas, de acuerdo con Moreno-Léon, Román-González y Robles (2017). Por una parte, se puede trabajar el refuerzo de conceptos teóricos más concretos. En este caso, las actividades son tipo puzles o retos guiados, en las que se trabaja de forma independiente cada concepto del pensamiento computacional, pero no de forma relacionada entre conceptos.

Por otra parte, en un segundo nivel, los alumnos proponen soluciones a retos más generales para la construcción de programas. Estas soluciones tienen que establecerse a partir del empleo de los procesos superiores del pensamiento computacional. En este caso, las actividades deben ser creativas y con un enfoque global, como la creación de prototipos.

Aprender a codificar resulta más sencillo en los últimos años gracias a los entornos de programación por bloques. Estos entornos de pseudocódigo facilitan la creación de programas al centrarse en la lógica de programación y no en las características de un lenguaje. Además de que puede utilizarse con múltiples fines.

En cuanto a las técnicas de instrucción, se encuentra el uso de proyectos o la resolución de problemas. La creación de un proyecto de código debe abarcar diferentes fases que implican desde el diseño inicial hasta las pruebas finales, fases que se ven facilitadas si se aplica un pensamiento computacional.

Yang, Swasnon, Chittoori y Baek (2018) proponen cómo trabajar los elementos básicos que componen un proyecto STEM de forma que se adopte una perspectiva computacional. En primer lugar, se debe elaborar una descripción del proyecto. En segundo lugar, hay que establecer una pregunta guía para la investigación, y dividir esta pregunta en preguntas más simples; las cuales corresponden al objetivo del problema y a su descomposición en subproblemas. Esta manera permite abordar un proyecto STEM con un enfoque de resolución de problemas.

Entre las temáticas más habituales para realizar proyectos de programación, un proyecto puede tener como objetivo crear un cuento animado, o Storytelling. Algunos entornos de programación por bloques están orientados a ello, como Alice (Werner, Denner, Bliesner y Rex, 2009) o Scratch (Burke y Kafai, 2012).

Otra temática atractiva es el diseño y desarrollo de videojuegos (Repenning, Webb y Ioannidou, 2010; Lee *et al.*, 2011), ya que se trata de proyectos complejos en los que debe aplicarse una estrategia computacional para abordar todas las mecánicas y dinámicas para que funcione.

La robótica es otra opción para la creación de proyectos, ya que además de codificar sus funcionalidades, deben tener en cuenta la parte física del proyecto, es decir, deben conocer todos los componentes electrónicos para determinar qué componentes necesitan y cómo se van a relacionar unos con otros. Este ámbito puede resultar más complejo de aplicar por la necesidad de adquirir más recursos,

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10
Cuestiones Pedagógicas, *2*(29), 2020, pp.130-144

pero existen bastantes propuestas para trabajar el pensamiento computacional a través de la construcción y programación de robots (Roscoe, Fearn, y Posey, 2014; Atmatzidou y Demetriadis, 2016; Ortega-Ruipérez y Asensio, 2018; Bers, González-González, y Armas-Torres, 2019).

Por otra parte, el pensamiento computacional se puede trabajar a través de la resolución de problemas de programación, de forma que se empleen estrategias que permitan el desarrollo de competencias digitales (Basogain-Olabe, Olabe-Basogain y Olabe-Basogain, 2015).

En este sentido, Ortega-Ruipérez y Asensio (2018) detallan una estrategia para asegurar que se emplea el PC al abordar la resolución de problema de programación. Para ello, se basan en la diferenciación de procesos implicados en la resolución de cualquier problemas y procesos exclusivos de lo que denominan la estrategia computacional.

En primer lugar se debe insistir en identificar las diferentes partes que pueden extraer, de forma que se aborden independientemente (descomposición). Además, deben aprovechar los elementos de programación que conocen para elaborar código de forma óptima, como instrucciones de control, o funciones (automatización). También deben intentar que las acciones que no dependan de otras puedan ejecutarse paralelamente (paralelismo). Por último, deben imaginarse la ejecución de cada paso para intentar comprobar si funcionarán y si encajan los resultados (simulación).

Para finalizar la didáctica del PC a través de la codificación, es importante tener en cuenta que se deben combinar las actividades propuestas en la medida de lo posible, y en función del nivel madurativo y desarrollo cognitivo de los aprendices. Por este motivo, tal y como proponían Moreno-Léon *et al.* (2017), se recomienda iniciarse con actividades guiadas que permitan reforzar los conceptos por separado; para, consecutivamente, empezar con la codificación, la creación de programas a través de la resolución de problemas, o la creación de proyectos de cualquier tipo, como proyectos STEM, historias animadas, videojuegos o robótica.

Desarrollo del PC de forma “desenchufada” o “desconectada”

Desarrollo del PC con juegos y actividades lúdicas

Por una parte, existen experiencias en las que se emplean juegos de mesa (Apostolellis *et al.,* 2014; Bayeck, 2018), ya que muchos juegos se deben abordar desde una perspectiva computacional para tener en cuenta todas las variables para progresar. Por ejemplo, Tsarava *et al.* (2017) describen cómo hacerlo con una búsqueda del tesoro.

También se ha visto el uso de juguetes Montessori como una introducción a este pensamiento en niños de infantil (Zapata-Ros, 2019). Aunque no se trabaje desde la descomposición de los juegos, se pueden abordar los procesos de forma independiente.

Otra opción para trabajar los procesos del pensamiento computacional por separado, a modo de introducción, sería con actividades lúdicas tipo retos, como puede ser el dibujar piezas de Tetris siguiendo unas instrucciones limitadas (Brackmann *et al.*, 2017), o de forma más dinámica a través de construir puzles, seguir pistas,…en una gymkana (Jagušt, Krzic, Gledec y Grgi, 2018).

Desarrollo del PC con resolución de problemas

La resolución de cualquier problema, descomponiendo el problema en otros más sencillos de abordar, también facilita el PC aunque no sea programando. Los procesos más fáciles de desarrollar en la resolución de problemas, estudiada desde una perspectiva computacional, son los relacionados con la creación de algoritmos, mientras que los procesos más complejos o difíciles de desarrollar son los que se relacionan con la representación del problema (Ortega-Ruipérez, 2018).

En este sentido, Ortega-Ruipérez y Asensio (2018), proponen el planteamiento de diferentes fases o niveles en cuanto al andamiaje necesario para afrontar retos o problemas desde una perspectiva computacional.

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10

En un primer nivel o fase, se debe acompañar a los alumnos durante todo el ejercicio de resolución, tanto en la representación del problema como en la creación de algoritmos que permitan resolverlo. Se recomienda que la representación del problema se realice a través de mapas conceptuales, diagramas de flujo o cualquier otro recurso que facilite la organización de la información. Debe abordarse de forma conjunta en el aula, explicando la representación del problema para observar los elementos que se pueden descomponen en problemas diferentes. Igualmente, las propuestas sobre los algoritmos se realizarían de forma conjunta, guiada por el profesor, asegurando que la representación es adecuada.

En un segundo nivel, dominarán cómo crear los algoritmos, por lo que el proceso guiado debe limitarse a la representación del problema, seguida por la creación de algoritmos de forma autónoma por el alumnado, aunque con la supervisión del profesor, atendiendo a los enfoques más óptimos para comentarlos con el grupo.

En la tercera fase, serán capaces de crear la representación y los algoritmos, aunque se recomienda que, antes de comenzar la algoritmización, se haga una puesta en común de la representación, y que tras esto el profesor concluya con la representación completa.

Desarrollo del PC mediante proyectos

Algunos de los proyectos que se plantean a través de programación, se pueden aplicar sin utilizar programación.

Por ejemplo, la creación de historias o Storytelling, ya que se aborda normalmente desde la descomposición de la historia en capítulos, viñetas,…o lo que requiera el formato utilizado, ya que puede trabajarse con varios formatos que no implican tecnología. Curzon, Mcowan, Plant y Meagher (2014) la utilizan en la formación de profesores.

Kordaki y Kakavas (2017) estudian los procesos del pensamiento computacional que se utilizan, y concluyen que sólo con la creación del storyboard, que corresponde a una representación gráfica del conjunto de pasos en los que se descompone la historia, se trabajan varios pensamientos (lógico, crítico, algorítmico,..) además de la descomposición y la representación de datos, por mencionar los más relevantes.

Otro ejemplo de proyectos de programación que pueden trabajarse de manera similar como un proyecto con otro formato no digital y que igualmente desarrollen el pensamiento computacional es el diseño de juegos, como los juegos de mesa. Igualmente deben descomponer el juego de forma general en las reglas, dinámicas, progreso, etc. que requiere para que sea jugable.

Conclusiones

En estas páginas se incluye un modelo teórico del pensamiento computacional que pretende servir para la comprensión de este pensamiento, a través de la creación de una definición operativa con un análisis desde la psicología de lo que se considera un pensamiento (Carretero y Asensio, 2004).

Se han obtenido cinco procesos que siempre están presentes cuando se emplea el pensamiento computacional: abstracción, evaluación, generalización, creación de algoritmos y descomposición, coincidiendo con la propuesta de otros autores (Looi *et al.*, 2018). Entre estos, la descomposición es la clave de este pensamiento (Ortega-Ruipérez, 2018) que posibilita que el resto de procesos se trabajen de forma simplificada, facilitando por tanto la abstracción, evaluación, generalización y creación de algoritmos de las diferentes partes descompuestas.

A partir de aquí se han propuesto directrices pedagógicas para abordar el PC, a través de programación, de juegos desconectados, de resolución de problemas y creación de proyectos, en base a la literatura y siguiendo el modelo creado, que pretende servir de guía para el diseño de actividades didácticas que persigan el desarrollo de este pensamiento.

ISSN 0213-7771 - e-ISSN 2443-9991
https://doi.org/10.12795/CP.2020.i29.v2.10
Cuestiones Pedagógicas, *2*(29), 2020, pp.130-144

Como limitación se puede destacar la poca literatura que trabaja el pensamiento computacional ajeno a la programación o codificación. Como prospectiva, se propone profundizar en cómo se puede trabajar el PC a través de actividades desconectadas, ya sean juegos, problemas o proyectos.

Referencias

Académie des Sciences. (2013, mayo). *Teaching computer science in France: Tomorrow can't wait.* https://bit.ly/3f9d54F

Adell, J., Llopis, M. A., Esteve, F. y Valdeolivas, M. G. (2019). El debate sobre el pensamiento computacional en educación. *RIED, 1*(22), 171–186. https://doi.org/10.5944/ried.22.1.22303

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J. y Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework : Implications for Teacher Knowledge. *Educational Technology & Society, 19* (3), 47–57. https://bit.ly/3381zUZ

Apostolellis, P., Tech, V., Stewart, M., Tech, V., Frisina, C., Tech, V., … Tech, V. (2014). *RaBit EscApe : A Board Game for Computational Thinking* [ponencia]. Proceedings of the 2014 Conference on Interaction Design and Children. https://bit.ly/3fbakzD

Atmatzidou, S. y Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75, 661–670. https://doi.org/10.1016/j.robot.2015.10.008

Barr, V. y Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *Acm Inroads, 2*(1), 48–54. https://doi.org/10.1145/1929887.1929905

Basogain-Olabe, X., Olabe-Basogain, M. Á. y Olabe-Basogain, J. C. (2015). Pensamiento Computacional a través de la Programación: Paradigma de Aprendizaje. *Revista de Educación a Distancia (RED)*, 46, 1–33. https://doi.org/10.6018/red/46/6

Bassok, M. y Novick, L. R. (2012). Problem solving. In K. J. Holyoak y R. G. Morrison (Eds.), *The Oxford Handbook of Thinking and Reasoning* (pp. 413–432). Oxford University Press.

Bayeck, R. Y. (2018). A review of five African board games: is there any educational potential? *Cambridge Journal of Education, 48*(5), 533–552. https://doi.org/10.1080/0305764X.2017.1371671

Bers, M. U., González-González, C. y Armas-Torres, M. B. (2019). Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers and Education*, 138, 130–145. https://doi.org/10.1016/j.compedu.2019.04.013

Bers, M. U. (2017). *Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom.* Routledge. https://doi.org/10.4324/9781315398945

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A. y Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education - Implications for policy and practice.* Joint Research Centre (JRC). https://doi.org/10.2791/792158

Bocconi, S., Chioccariello, A. y Earp, J. (2018). *The Nordic Approach To Introducing Computational Thinking And Programming In Compulsary Education.* Report prepared for the Nordic@BETT2018 Steering Group. https://doi.org/10.17471/54007

Brackmann, C. P., Román-gonzález, M., Robles, G., Moreno-león, J., Casali, A. y Barone, D. (2017). *Development of Computational Thinking Skills through Unplugged Activities in Primary School* [ponencia]. Proceedings of the 12th Workshop on Primary and Secondary Computing Education. https://doi.org/10.1145/3137065.3137069

Brennan, K. y Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking* [ponencia]. Annual American Educational Research Association meeting AERA, Canadá. https://bit.ly/39BynXj

Burke, Q. y Kafai, Y. B. (2012). *The Writers' Workshop for Youth Programmers. Digital Storytelling with Scratch in Middle School Classrooms* [ponencia]. SIGCSe: Proceedings of the 43rd ACM technical symposium on Computer Science. https://doi.org/10.1145/2157136.2157264

Carretero, M. y Asensio, M. (2004). Introducción. *Psicología del pensamiento.* Alianza.

Carretero, M. y Asensio, M. (2008). *Psicología del pensamiento* (2a ed.). Alianza.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. y Woollard, J. (2015). *Computational thinking A guide for teachers.* CAS: Computer At School. https://bit.ly/2X0YzWA

Curzon, P., Mcowan, P. W., Plant, N. y Meagher, L. R. (2014). *Introducing Teachers to Computational Thinking Using Unplugged Storytelling* [ponencia]. WiPSCE '14: Proceedings of the 9th Workshop in Primary and Secondary Computing https://doi.org/10.1145/2670757.2670767

Delcker, J. y Ifenthaler, D. (2017). Computational Thinking as an Interdisciplinary Approach to Computer Science School Curricula: A German Perspective. En Rich, P. y Hodges, C. (eds.) *Emerging Research, Practice, and Policy on Computational Thinking. Educational Communications and Technology: Issues and Innovations.* Springer. https://doi.org/10.1007/978-3-319-52691-1

Dörner, D. y Funke, J. (2017). Complex problem solving: What it is and what it is not. *Frontiers in Psychology,* 8, 1–11. https://doi.org/10.3389/fpsyg.2017.01153

Duncan, G. J., Magnuson, K. y Votruba-Drzal, E. (2014). Boosting family income to promote child development. *Future of Children, 24*(1), 99–120. https://doi.org/10.1353/foc.2014.0008

Fischer, A., Greiff, S. y Funke, J. (2017). The history of complex problem solving. En Csapó, B. y Fune, J. (Eds) *The nature of problem solving.* Centre for Educational Research and Innovation. https://doi.org/10.1787/9789264273955-9-en

Funke, J., Fischer, A. y Holt, D. (2017). When Less Is Less: Solving Multiple Simple Problems Is Not Complex Problem Solving—A comment on Greiff *et al.* (2015). *Journal of Intelligence, 5*(1), 5. https://doi.org/10.3390/jintelligence5010005

González-González, C. S. (2018). La enseñanza-aprendizaje del Pensamiento Computacional en edades tempranas: una revisión del estado del arte. En Zapata-Ros, M. y Villalba-Condor, K.O. (eds.), *Pensamiento Computacional*. https://bit.ly/3f8mDwC

Gretter, S. y Yadav, A. (2016). Computational Thinking and Media & Information Literacy: An Integrated Approach to Teaching Twenty-First Century Skills. *TechTrends*, 60, 510-516. https://doi.org/10.1007/s11528-016-0098-4

Grover, S. y Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, 42(1), 38–43. https://doi.org/10.3102/0013189X12463051

Herde, C. N., Wüstenberg, S. y Greiff, S. (2016). Assessment of Complex Problem Solving: What We Know and What We Don't Know. *Applied Measurement in Education, 29*(4), 265–277. https://doi.org/10.1080/08957347.2016.1209208

ISTE (2016). *ISTE standards for students.* https://bit.ly/3gamV7y

ISTE y CSTA (2011). *Computational Thinking Leadership Toolkit*. https://bit.ly/39B7HX2

Jagušt, T., Krzic, A. S., Gledec, G. y Grgi, M. (2018). *Exploring Different Unplugged Game-like Activities for Teaching Computational Thinking* [ponencia]. IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/FIE.2018.8659077

Kong, S.-C. y Abelson, H. (2019). Computacional Thinking Education. En Kong, SC y Abelson, H. (eds) *Computational Thinking Education*. Springer Link. https://doi.org/10.1007/978-981-13-6528-7

Kordaki, M. y Kakavas, P. (2017). *Digital Storytelling As an Effective Framework for the Development of Computational Thinking Skills* [ponencia]. 9th International Conference on Education and New Learning Technologies. https://doi.org/10.21125/edulearn.2017.2435

Kwisthout, J. (2012). Relevancy in Problem Solving: A Computational Framework. *The Journal of Problem Solving, 5*(1), 18–33. https://doi.org/10.7771/1932-6246.1141

Lee, I., Martin, F., Denner, J., Coulter, B., Walter, I., Erickson, A. J., … Werner, L. (2011). Computational Thinking for Youth in Practice. *ACM Inroads, 2*(1). https://doi.org/10.1145/1929887.1929902

Looi, C.-K., How, M.-L., Longkai, W., Seow, P., y Liu, L. (2018). Analysis of linkages between an unplugged activity and the development of computational thinking. *Computer Science Education, 28*(3), 255–279. https://doi.org/10.1080/08993408.2018.1533297

Moreno-Léon, J., Román-González, M. y Robles, G. (2017). Programar para aprender en Educación Primaria y Secundaria : ¿qué indica la evidencia empírica sobre este enfoque? *ReVisión, 10*(2).

Ortega-Ruipérez, B. (2018). *Pensamiento computacional y resolución de problemas* [Tesis doctoral, Universidad Autónoma de Madrid]. Repositorio institucional UAM. https://bit.ly/2P3iPCk

Ortega-Ruipérez, B. y Asensio, M. (2018). Robótica DIY: pensamiento computacional para mejorar la resolución de problemas. *RELATEC, 17*(2), 129–144. https://doi.org/10.17398/1695-288X.17.2.129

Repenning, A., Webb, D. y Ioannidou, A. (2010). *Scalable game design and the development of a checklist for getting computational thinking into public schools* [ponencia]. SIGCSE'10 - Proceedings of the 41st ACM Technical Symposium on Computer Science Education. https://doi.org/10.1145/1734263.1734357

Román-González, M., Pérez-González, J. C. y Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, 72, 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Roscoe, J. F., Fearn, S. y Posey, E. (2014). *Teaching computational thinking by playing games and building robots* [ponencia]. Proceedings - 2014 International Conference on Interactive Technologies and Games, ITAG 2014. https://doi.org/10.1109/iTAG.2014.15

Selby, C. y Woollard, J. (2013). *Computational Thinking : The Developing Definition* [ponencia]. ITiCSE Conference, England. https://bit.ly/3hLA1Zg

Shute, V. J., Sun, C. y Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Smith, E. E. y Kosslyn, S. M. (2008). C*ognitive psychology. Mind and Brain.* Pearson Hall.

Tsarava, K., Moeller, K., Pinkwart, N., Butz, M., Trautwein, U., Ninaus, M. y Institut, L. (2017). *Training Computational Thinking : Game-Based Unplugged and Plugged-in Activities in Primary School* [ponencia]. 11th European Conference on Game-Based Learning ECGBL. Austria.

Valverde-Berrocoso, J., Fernández-Sánchez, M. R. y Garrido-Arroyo, M. C. (2015). El pensamiento computacional y las nuevas ecologías del aprendizaje. *Revista de Educación a Distancia (RED), 46*(46). https://doi.org/10.6018/red/46/3

Voogt, J., Fisser, P., Good, J., Mishra, P. y Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715–728. https://doi.org/10.1007/s10639-015-9412-6

Vuorikari, R., Punie, Y., Carretero, S. y Van Den Brande, L. (2016). *DigComp 2.0: The Digital Competence Framework for Citizens.* Publications Office of the European Union https://doi.org/10.2791/11517

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Kemi, J., Trouille, L. y Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology, 25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Werner, L., Denner, J., Bliesner, M. y Rex, P. (2009). *Can middle-schoolers use Storytelling Alice to make games? Results of a pilot study* [ponencia]. FDG 2009 - 4th International Conference on the Foundations of Digital Games, Proceedings. https://doi.org/10.1145/1536513.1536552

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1201/b16812

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. M. (2011). Research Notebook: Computational Thinking--What and Why? *The Link. The Magazine of Carnegie Mellon University's School of Computer Science*. https://bit.ly/3hK39QY

Wing, J. M. (2014). *Computational thinking benefits society* [ponencia]. 40th Anniversary Blog of Social Issues in Computing. https://bit.ly/2P1wkmf

Wing, J. M. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology, 25*(2), 7–14. https://doi.org/10.17471/2499-4324/922

Yang, D., Swasnon, S. R., Chittoori, B. B. C. y Baek, Y. (2018). *Work in progress: Integrating computational thinking in STEM education through a project-based learning approach* [ponencia]. ASEE Annual Conference and Exposition, Conference Proceedings. https://bit.ly/3f6lWUE

Zapata-Ros, M. (2015). Pensamiento computacional: Una nueva alfabetización digital. *Revista de Educación a Distancia (RED), 46*(15). https://doi.org/10.6018/red/46/4

Zapata-Ros, M. (2019). Computational Thinking Unplugged. *Education in the Knowledge Society,* 20, 1–29. http://dx.doi.org/10.14201/eks2019_20_a18