RSEUS ISSN: 2393.7815
REVISTA SUDAMERICANA DE EDUCACIÓN, UNIVERSIDAD Y SOCIEDAD

AÑO 9, NÚMERO 1
https//doi.org/10.48163/rseus.2021.9145-59



# Pensamiento computacional, educación STEM y la educación informática: cuestiones pendientes

# Computational Thinking, STEM Education and Informatics Education: Pending Issues

**Recibido: 25/11/2021**
**Aprobado: 15/12/2021**

Manuela Cabezas[1]

## Resumen

El artículo examina el auge de la expresión Pensamiento Computacional (PC) en el campo educativo, desde la perspectiva de las transformaciones producidas por la computación en los últimos 20 años y su impacto en la educación. Prestando particular atención a dicho contexto, se cuestionan las iniciativas educativas para el PC como solución al problema educativo para la educación de hoy y se establecen lineamientos para el abordaje didáctico del PC desde los aportes de la didáctica de la programación. El objetivo es problematizar el enfoque didáctico del PC desde la perspectiva de la Educación Informática como base para el modelo didáctico de las ciencias computacionales.

*Palabras clave*: educación informática; pensamiento computacional; STEM; didáctica

## Abstract

This paper examines the rise of the phrase Computational Thinking (CT) in the field of education, from the perspective of the computational revolutions of the last 20 years and their impact on education. Paying special attention to said context, the paper challenges educational initiatives for basic CT as the solution for the educational problems of today and sets guidelines for the didactic approach for CT grounded on the theoretical foundations of the didactics of programming. The purpose of the paper is to contribute to the development of a didactic model for computational science from the discipline of Informatics Education.

*Keywords:* informatics education, computational thinking, STEM, didactics

[1] Doctora en Educación. Universidad de la Empresa, Coordinadora de la Maestría en Educación, Facultad de Ciencias de la Educación, Montevideo, Uruguay. mcabezas@ude.edu.uy

## Introducción

Es un hecho ampliamente reconocido que en los últimos 30 años la computación ha transformado la ciencia, el conocimiento y la sociedad en general de manera fundamental (Castells, 2000; Denning y Tedre, 2019). Desde fines de la década de los 90 la educación se encuentra en el foco de la atención política que busca transformar la educación en respuesta a las necesidades en la sociedad del conocimiento y las industrias emergentes y nuevos mercados laborales (Atkin y Black, 2005).

La educación y su relación con los cambios tecnológicos ha sido objeto de crítica desde hace muchas décadas. En particular se critican actitudes conservadoras que caracterizan a las instituciones, prácticas y actores de la educación con relación al cambio y la innovación en general (Nicholls, 1983), y a las tecnologías digitales en particular (Henderson y Romeo, 2015; Serdyukov, 2017).

Sin embargo, en los últimos años el área de la tecnología educativa, o edTech, atrae un fuerte interés de múltiples actores, industrias y sectores que están presentes en la educación con una plétora de soluciones digitales (apps, media, redes, juegos, etc.) (Komljenovic, 2021). El área nace en la década de los 80 con gran entusiasmo en torno al potencial de la tecnología como solución a varios problemas complejos del campo educativo (Koschmann, 1996; Sancho-Gil, Rivera-Vargas, y Miño-Puigcercós, 2020).

Si bien muchas de las ilusiones generadas desde la academia no se materializaron, años después el sector revive con nuevas expectativas en torno al uso de ciencia de datos aplicada a la educación (modelado, analítica de aprendizaje, sistemas de aprendizaje adaptativo, etc.) y un renovado interés por parte de las corporaciones Tech (Roberts-Mahoney y Garrison, 2016; Sancho-Gil et al., 2020; Williamson, 2017). En efecto, en los últimos años, el sector se ha convertido en un dinámico espacio de inversión e innovación que experimenta un crecimiento exponencial abarcando múltiples sectores y actores públicos y privados, que al 2020 viene atrayendo inversiones que sobrepasan los 16 billones de dólares (Komljenovic, 2021). Varios autores entienden que estos cambios representan una reorganización de la educación o, en las palabras de Decuypere, Grimaldi, y Landri, (2021), una plataformización de la educación, haciendo referencia al crecimiento del sector edTech, la multiplicidad de actores involucrados (privados y cuasi públicos) y las reformas educativas a pequeña, mediana y gran escala (Cone et al., 2021; Decuypere et al., 2021; Sancho-Gil et al., 2020). A esto debemos sumarle la llegada de la pandemia del COVID-19 (SARS-CoV-2) y la respuesta política en situación de emergencia, con la cual los procesos de digitalización y datificación de la educación (Mayer-Schönberger y Cukier, 2013; Selwyn, 2013) que ya estaban presentes en la educación, se vieron fuertemente acelerados y profundizados a nivel global (Cone et al., 2021; UNESCO, 2020).

Este artículo surge en el marco del grupo de trabajo interdisciplinario de las Facultades de Ingeniería y Ciencias de la Educación de la Universidad de la Empresa (UDE), que explora el rol de la informática en varias áreas y espacios de educación e innovación. Una de las áreas de investigación son las didácticas STEM(p) que examina el papel de la Educación Informática (EI) para la educación del futuro. En particular, se

consideran las dificultades que enfrenta la EI como disciplina, con relación a otros proyectos educativos como lo son la educación STEM y el Pensamiento Computacional (PC). El objetivo del artículo es, por un lado, brindar argumentos que permitan esclarecer la situación de la EI en el contexto educativo actual y, por otro lado, presentar aportes para un modelo didáctico para las ciencias computacionales desde la didáctica de la programación, al que denominamos modelo STEM(p). En el artículo se utiliza EI para la educación en ciencias de la computación, y CS para la ciencia de la computación, por su sigla en inglés.



## La Educación Informática y STEM

Si consideramos el estado de la educación de la disciplina CS, podemos hablar de una educación informática a partir de los últimos 20 años, ya que anteriormente y hasta fines de la década del 90 las contribuciones académicas al área se limitaban en gran parte al desarrollo de herramientas, propuestas conductistas (aprendizaje asistido por computadora, CAI) por un lado, y descripciones de cursos, intercambio de experiencias y otras investigaciones no empíricas, por el otro (Koschmann, 1996; Tedre, 2020). Entendemos que el campo como tal se desarrolla y profesionaliza principalmente después del año 2000 (Tedre, 2020), en gran parte desde los debates e investigaciones de la educación universitaria, particularmente en los cursos de grado (ver Nwana, 1997). Durante gran parte de la primera década del 2000 los esfuerzos del área estuvieron centrados en combatir la confusión existente sobre cuáles son, o no, los dominios y límites de la disciplina con relación al proceso de enseñanza y aprendizaje (Berry et al., 2013; Dowek, 2005; Holmboe, McIver, y Carlisle, 2001; Nwana, 1997). Fue necesario, por ejemplo, distinguir la EI del uso de la tecnología como herramienta de apoyo para el trabajo pedagógico para el aprendizaje de otras disciplinas e introducir el problema didáctico desde la investigación en la educación de CS (Dowek, 2005; Holmboe et al., 2001). Durante estos años varias investigaciones empíricas sobre el aprendizaje de la programación aportan al esfuerzo por desarrollar los fundamentos teóricos desde planteamientos pedagógicos para la didáctica de la disciplina (da Rosa, 2004; Gomes, 2007; Götschi y Galpin, 2003; Hubwieser, P., Armoni, M., Giannakos, M. y Mittermeir, R., 2014; Lister, 2011; Mannila y Salakoski, 2007; Saeli, Perrenet, Jochems, y Zwaneveld, 2011; Schwill, 1997). Sin embargo, la confusión sobre el lugar que ocupa la disciplina como ciencia, no ayudó a la hora de integrar adecuadamente la computación al sistema educativo preuniversitario; no como herramienta o tecnología, sino como disciplina científica (Dowek, 2005; Holmboe et al., 2001). La confusión existente en torno a la disciplina está indudablemente ligada también a los extensos debates internos sobre la naturaleza de la disciplina y sus fundamentos. Si bien la computación es una ciencia sumamente exitosa, también es una disciplina que ha sufrido varias crisis de identidad (Nwana, 1997). Una disciplina que no se deja encapsular fácilmente implica mayores dificultades a la hora de consolidar un currículo y lineamientos para su educación. Especialmente a nivel preuniversitario, los sistemas y modelos educativos representan una verdadera dificultad que ha marcado la educación de la CS desde sus comienzos y, hasta el día de hoy, todavía estamos lejos de hablar de una educación de la computación en primaria o secundaria a la par de otras ciencias (CECE, 2017).

Por otro lado, es notorio que es durante este mismo período que surge la reforma de la educación bajo el acrónimo STEM, principalmente en Estados Unidos,

Reino Unido y Europa (denominado así por la National Science Foundation en los años 90). Aquí hay un punto importante a resaltar desde el punto de vista de la evolución de la EI y la educación STEM. A diferencia de la EI, que surge del propio proceso para establecer una disciplina académica, la educación STEM es producto directo de una agenda política que busca responder a los cambios en el mercado laboral y las tendencias negativas detectadas en los indicadores económicos durante la década de los 90 (Sanders, 2009; Williams, 2011). Países como Estados Unidos declaran a fines de la década de los 90 la crisis del conocimiento como la mayor amenaza al dominio económico y político global de occidente, con lo cual se desata una era de reformas educativas con énfasis en formar especialistas para las nuevas industrias tecnológicas (Blackley y Howell, 2015; Williams, 2011).

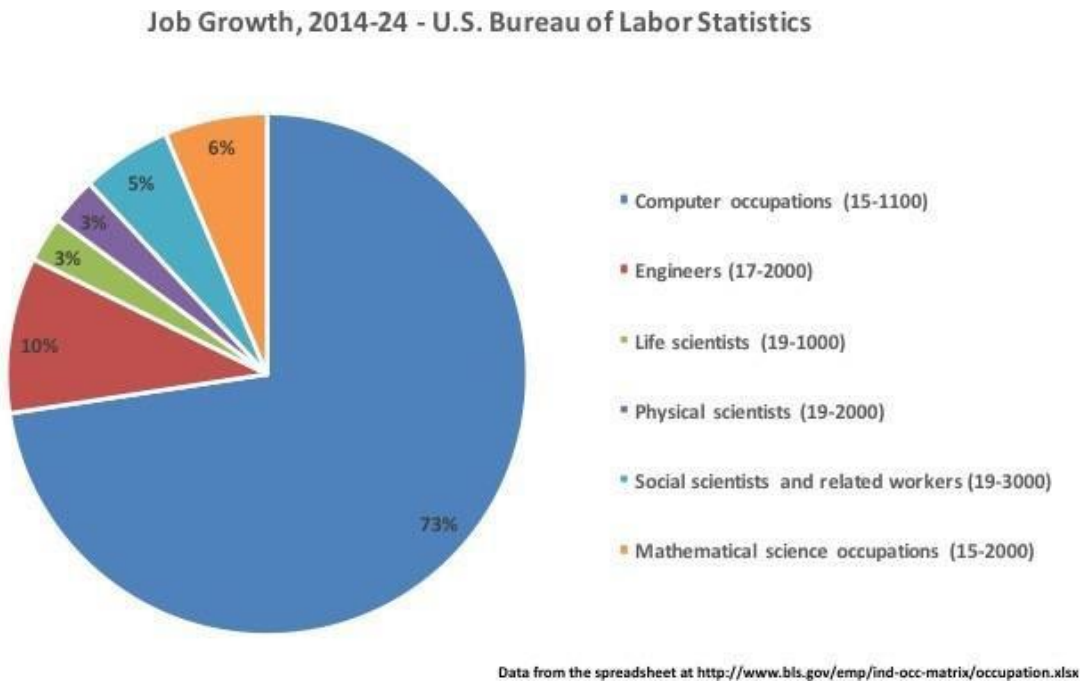
Al igual que la campaña por la educación informática, los primeros años de las reformas para la educación STEM fueron poco fructíferos en ese sentido, ya que las necesidades de fomentar y aumentar el número de especialistas en los campos de la ingeniería y las áreas tecnológicas difícilmente podían ser atendidos por sistemas educativos en los que dos de las cuatro letras del STEM (T y E) no tenían un currículo ni marco definido para su abordaje (Blackley y Howell, 2015). Esto llevó a que el campo de la educación STEM se desarrollara primordialmente dentro de las áreas que ya existían en el currículo y en la formación docente: la S de las ciencias (física, biología, química) y M de la matemática (Blackley y Howell, 2015).

Pasada una década de reformas STEM, vemos la publicación en 2010 del informe “Running on Empty” (Wilson, Sudol, Stephenson, y Stehlik, 2010), en el cual la Association for Computing Machinery (ACM) junto con la Asociación de Docentes en CS (CSTA) explica las dificultades que enfrenta la disciplina informática en el sistema educativo, la profunda confusión que existe a nivel de terminología, la exclusión de la disciplina en muchos de los sistemas formales de certificación (sus créditos se cuentan como créditos en matemática u otra disciplina), la poca acreditación desde los conocimientos del área, y la falta de estándares en el currículo y apoyo a la formación docente. Es interesante ver cómo el reporte que se autodefine como parte de la “era de la educación STEM” deja en evidencia las barreras institucionales que afectan la educación informática, llegando a preguntar en la página 30: “¿Dónde está la Educación Informática en STEM?” y sobre la exclusión de la computación en STEM, señala: “Generalmente, [la computación] no es ni explícita ni sutilmente parte de los cursos STEM. Y la evidencia de este problema es clara. [...] Muchos observadores creen que computación es la T en STEM, pero en gran medida, este no es el caso” (Wilson et al., 2010, p. 30).

Vale destacar que apenas unos años después de la publicación del reporte de la CSTA, la agencia de estadística laboral de Estados Unidos (BLS) publicara la proyección para los trabajos en las áreas STEM para los próximos 10 años (2014-2024), que se muestra en la figura 1, donde se ve claramente que el mercado laboral STEM es informático (73% de los trabajos en las áreas STEM son del área de la computación).

**Figura 1**

*Proyección para los trabajos en las áreas STEM para 2014-2024*




Nota: Tomado de *Agencia de estadística laboral de Estados Unidos. http://www.bls.gov/emp/ind- occ-matrix/occupation.xlsx*

Apenas unos años después, ACM Europe e Informatics Europe publicaron el informe “Informatics for all” donde piden el reconocimiento de la disciplina como ciencia básica para la educación del siglo 21 (Informatics for All, 2018). El informe, publicado en 2018, en pre- pandemia, plantea que el dominio de los sistemas digitales “opera bajo leyes científicas, al igual que lo hacen los de las máquinas físicas y los cuales conforman la ciencia denominada “informática”. Agregan que, si hoy el mundo es digital, es lamentable que la ciencia que sostiene ese mundo, y que es una disciplina científica definida por sus propios conceptos, métodos y cuerpo de conocimiento, aun no es reconocida como ciencia en los sistemas educativos, ni es priorizada en la educación. (Informatics for All, 2018). El informe hace referencia a los informes que vienen evaluando la situación de la disciplina en sistemas educativos formales (CECE, 2017; Wilson et al., 2010).

## El paradigma de las ciencias computacionales y del pensamiento computacional

En los últimos 10 años, mientras la EI continúa su campaña para posicionarse ante las políticas de la educación para el “mundo digital”, emerge en el campo educativo la expresión Pensamiento Computacional (PC). La expresión se populariza como propuesta educativa que genera un gran impacto entre académicos, docentes, maestros, escuelas, gestores y directores de institutos (Denning y Tedre, 2016).

¿Cómo se explica la explosión global del PC en un sistema educativo en el que la incorporación de la informática ha resultado tan difícil, ya sea en los sistemas educativos o en los modelos STEM?

Justamente, entendemos que el éxito del PC representa una respuesta política a la necesidad de integrar la revolución computacional en la educación de forma desconectada de la disciplina CS y su educación, la cual continúa en vías de desarrollo (ver Kong y Abelson, 2019). Sin embargo, existe un cuestionamiento sobre las políticas educativas que implican un importante gasto público y prometen una educación para "la sociedad del conocimiento y la información", en gran parte sin evidencias empíricas ni fundamentos teóricos sólidos (Selwyn 2015, Denning 2017), y muchas veces sin atender las necesidades de la comunidad educativa (Decuypere et al., 2021; Sancho-Gil et al., 2020). Partiendo de esta crítica, si bien no negamos la necesidad de poner al alcance de nuestros estudiantes las herramientas necesarias para enfrentar los desafíos de la era digital, entendemos que los esfuerzos por innovar en la educación sin el reconocimiento y consolidación de la EI no resolverá los desafíos que nos presenta el mundo digital. Para explicar este argumento vamos a definir más detalladamente a qué nos referimos con "la sociedad del conocimiento y la información" y por qué entendemos que el abordaje del PC debe partir de la EI.



En su revisión del impacto de la computación en las ciencias, Peter Denning y Matti Tedre (2015) describen cómo la computación ha producido dos revoluciones. La primera, un cambio radical en las prácticas científicas, es el producto de la informática como herramienta aplicada a las ciencias, debido al insuperable potencial y versatilidad de la computación y la simulación.

La segunda revolución, consiste en considerar el computar como una forma completamente nueva de ver los fenómenos naturales y artificiales, cambiando fundamentalmente la forma en que otros campos se ven a sí mismos y hacen su trabajo. Lo que hoy se llama ciencia computacional implica una nueva era de la ciencia (Denning y Tedre, 2015; 2021). Las transformaciones que surgieron al poner las herramientas de la ciencia de la computación al servicio de otras ciencias fueron abriendo nuevos problemas, preguntas y campos de investigación, porque, como explican Denning y Tedre, "Cuando la ciencia se vuelve más computacional, los límites de la computación trazan nuevos límites para el conocimiento" (Denning y Tedre, 2021, p. 21). De hecho, las ciencias del siglo 21 aportan rigor y orden a sus conceptos y teorías desde prácticas computacionales de modelado y simulación (Denning, 2017). Las ciencias computacionales integran la computación no solo para dar apoyo a teorías y experimentos tradicionales, sino también para ofrecer formas revolucionarias de interpretar procesos naturales y conducir investigaciones científicas (Denning, 2017; Markowetz, 2017). En las palabras de Markowetz (2017), "Hoy en día toda biología es biología computacional". Este contexto fue denominado como el "paradigma computacional de la ciencia" por Matti Tedre en la decimoprimera cumbre de ECSS (ECSS 2015, 11th Summit of Informatics Europe).

La velocidad de los cambios es notoria también en los departamentos e institutos de computación que, en los últimos años, se ven inundados de estudiantes de otras disciplinas que buscan desarrollar sus competencias computacionales para tener buen desempeño en sus carreras (Camp et al., 2017).

Si bien la computación como disciplina nace desde la interdisciplinaridad, el paradigma computacional de la ciencia (PCC) se caracteriza por la multi y

transdisciplinariedad que se ve reflejada tanto en las publicaciones científicas e industrias tecnológicas emergentes, como en las especializaciones que las producen: ciencia de datos, bioinformática, biología de sistemas, química computacional, etc. (Bibri, 2021; Hazra, Singh, Goyal, Adhi-kari, y Mukherjee, 2019).

El esfuerzo por reformular las ciencias como ciencias computacionales se viene realizando desde las comunidades académicas en las propias carreras e institutos, que están buscando desarrollar estrategias para acercar el PC a sus estudiantes de grado, en cursos que integran conceptos y métodos computacionales, desde la programación, modelado y simulación y las aplicaciones de analítica de datos (Aikat et al., 2017; Clauss y Nelsen, 2019; Mittal, Durak y Ören, 2017; Pollock, Mouza, Guidry, y K., 2019; Sharma, 2017; Sharma y Asirwatham, 2019).



Para la educación básica el desafío también se hizo presente, y es en este contexto que vemos el auge del pensamiento computacional de los últimos años (Barr y Stephenson, 2011; Weintrop et al., 2015). Podemos entender la respuesta a la idea del pensamiento computacional como la solución al problema de trasladar la transformación de las ciencias en ciencias computacionales, a la educación básica (Lodi y Martini, 2021). En otras palabras, cuando una idea resuena de forma desproporcionada en un contexto dado, muchas veces la explicación está en el contexto, y no en la originalidad o genialidad de la idea misma. En efecto, el fenómeno del PC fue recibido con tal entusiasmo en la educación que la comunidad de CS respondió con sorpresa y preocupación (ver Paulson, 2017).

Para entender mejor el fenómeno del PC en la educación, es interesante mirar con mayor atención cuál fue la idea que tanto se popularizó. La explosión de la expresión ocurre particularmente después de la publicación del artículo de Janette Wing en 2006, en el que define el PC como actitudes y habilidades de aplicación universal, y utilizando la abstracción y descomposición para enfrentar problemas complejos con la mentalidad de un científico de la computación (Wing, 2006). La aceptación de sus ideas fue prácticamente inmediata, generando un debate en los años siguientes, al cual la autora aportó definiciones complementando su idea inicial (Wing, 2008, 2010). Si bien la definición de Wing parte de la fundamentación de Papert sobre la transferibilidad de las destrezas de un científico en computación a la resolución de problemas en otros dominios (Papert, 1980), la definición inicial de Wing introduce un concepto fundamental: la idea de que el PC puede ser desarrollado desde el dominio de competencias generales y sin el uso del computador (Lodi y Martini, 2021).

Por supuesto que desde la publicación del artículo de Wing, las definiciones de PC varían ampliamente (incluyendo la de la propia Wing), lo cual dificulta la operacionalización del PC en la práctica (Cansu y Cansu, 2019). Sin embargo, podemos ver que las propuestas que presentan el PC como una forma de pensar y resolver problemas, independiente de los conceptos fundamentales de la computación, y como competencia fácilmente trasladable de un dominio a otro, resonaron profundamente en muchos educadores que vieron la posibilidad de introducir rápidamente el PC en sus aulas (Lodi y Martini, 2021).

Este enfoque del PC básico presenta requisitos de ingreso bajo, lo cual permite que maestros, docentes, y estudiantes tengan sus primeros acercamientos al PC, sin necesidad de manejar demasiados conceptos de computación y sin exigencias de conocimientos previos en programación (Denning y Tedre, 2021).

Desde la comunidad académica se ha expresado desde hace años la preocupación por la forma en que esta visión limitada reduce el PC a algunos componentes básicos que, además de ser comunes a todas las ciencias, ofuscan la relación algoritmo-máquina que es la base de la computación como disciplina (da Rosa, 2018; Denning y Tedre, 2015). Más específicamente, según Denning y Tedre (2021), la idea de definir la noción de algoritmo como una serie de pasos posiblemente ambiguos que se resuelven por humanos que computan (PC básico), es una conceptualización errónea de la computación, que omite la formación de varios conceptos claves de la computación que queremos introducir en la formación de los estudiantes. En las palabras de Pears (apud Pears, Tedre, Valtonen, y Vartiainen, 2021a): “el aprendizaje del PC tal como se define actualmente no permite que los niños comprendan cómo opera el mundo virtual.” El PC básico trabaja conceptos que no aportan a la conceptualización del funcionamiento de sistemas complejos, el manejo de la informática y el reconocimiento de fenómenos emergentes, todos componentes que necesitamos para entender el mundo en el PCC (Pears et al., 2021a). Muchos de los elementos claves que constituyen las competencias que buscamos desarrollar se diluyen al punto de perder sentido, y el desarrollo del PC como competencia compleja queda relegado al eventual momento en que los estudiantes encaran estudios universitarios (Denning y Tedre, 2015; Lodi y Martini, 2021; Pears et al., 2021a).

## STEM(p): didáctica para las ciencias computacionales

¿Cómo podemos desarrollar el PC desde los conceptos, métodos y cuerpo de conocimiento de la computación? Quizás sea momento de aclarar que al igual que Papert y Wing, partimos del potencial inherente en la transferibilidad del PC a otros dominios. La diferencia está en que el destino ya no son otras ciencias y dominios no computacionales, porque hoy la computación es la forma en que construimos conocimiento y entendemos el mundo, independientemente de qué ciencia estemos hablando. Lo que sí tenemos que definir es cuál es la tarea de la Educación informática como disciplina y cuál es el camino de la enseñanza y aprendizaje de las competencias de la computación.

En este sentido, en el campo de la educación se reconoce desde hace ya varias décadas, la necesidad de que la experiencia de aprendizaje sea autentica, introduciendo problemas/contextos reales y procesos que reflejan los procedimientos aplicados en las distintas disciplinas (Palm, 2008; Roach y Mitchell, 2018; Roth, Van Eijck, Reis, y Hsu, 2019). Es así que resulta natural mirar hacia las competencias y destrezas de los científicos y profesionales de hoy para definir la relevancia de lo que enseñamos y lo que queremos introducir a los estudiantes. Sin embargo, es importante distinguir lo que son las diversas áreas en las que se desempeñan los profesionales y los procesos y prácticas que les permitió desempeñarse en ellas.

Entonces, para empezar, es necesario aclarar el papel de la programación en la EI ya que el tema continúa pendiente. Aquí compartimos la opinión de varios autores que señalan que CS no es sólo programación (Armoni, 2016; Denning et al., 1989).

Sin embargo, en publicaciones recientes vemos que la programación continúa produciendo cierta confusión sobre el por qué y cómo enseñarla. Por ejemplo, en (Pears, Tedre, Valtonen, y Vartiainen, 2021b) los autores plantean que la dominancia

del paradigma imperativo ha sido más un problema que una solución y critican la manera en que se ha introducido la enseñanza de programación en la educación. Señalan que estamos ensenando CS del pasado y no del futuro y que *"We are currently at a point where it is legitimate to ask whether any knowledge about how to implement data structures and the algorithms to control them is any longer a central skill for most computing professions."* (Pears et al., 2021b).

Sin embargo, entendemos que la confusión surge del intento de transferir las destrezas y competencias provenientes de la vida profesional en áreas STEM, y no necesariamente del problema didáctico que aborda y permite el desarrollo del PC. En efecto, el cuestionamiento que plantean los autores en cuanto a la relevancia de los conocimientos vinculados al paradigma imperativo se refiere a la relevancia de dicha destreza para la vida profesional, donde los procesos de desarrollo de software se centran en seleccionar, convertir, recodificar y visualizar los datos para soluciones de "machine learning" que se ejecutan en "*cloud services*" donde los programas se encuentran encapsulados en herramientas cada vez más sofisticadas y de más alto nivel de abstracción (Pears et al., 2021b).

En cuanto a la enseñanza de programación, a pesar de que los autores recalcan su crítica al uso del paradigma imperativo, reconocen que *"It is not our intention to say that programming (or programming education) is becoming obsolete. Programming is a crucial skill for a range of jobs, and it will be a great asset in many fields. It will provide a good basic vocabulary for technology education"* (Pears et al., 2021b).

Los autores pasan así de considerar lo que los estudiantes deberían aprender en sus primeros contactos con la ciencia de la computación, a considerar las destrezas que necesita un profesional, reconociendo que hay una brecha entre ambas cosas, pero sin explicitar cómo superarla, que es, en nuestra opinión, la principal tarea de la EI del momento.

Por el contrario, lo que plantemos aquí desde las contribuciones a la didáctica de la programación de (da Rosa, Viera, y García-Garland, 2020), se enfoca en la base didáctica de lo que significa programar y su relación con el desarrollo del PC.

Entendemos que programar no es hacer que un programa funcione por medio de modificar el código hasta lograr el resultado deseado, sin analizar las razones que producen los errores, desgraciadamente una práctica bastante extendida, que llevó también a los autores en (Armoni, 2016; Pears et al., 2021b) a enfatizar que programar no es codificar.

Programar significa formular el problema como un problema algorítmico, esto es, determinar el conjunto de datos que conforman la entrada, especificar el resultado que se quiere obtener a partir de los mismos y diseñar una solución, es decir un algoritmo que transforme los primeros en lo segundo. La implementación del algoritmo en un programa, por otra parte, permite experimentar el significado de la abstracción en todo su potencial, al pasar de una solución de una instancia concreta de un problema a un programa que soluciona el problema general que puede ser ejecutado para (casi) cualquier número de casos.

¿Cómo se traslada esto al aprendizaje de los conceptos, métodos y competencias de la computación o, en otras palabras, qué significa realmente aprender a programar, en lugar de codificar?

Varios autores del área de la didáctica de la programación ilustran un proceso por el cual el estudiante reflexiona sobre las razones por las cuales un programa funciona (o no) y busca soluciones alternativas más eficientes. De esta forma el estudiante completa un proceso por el cual construye conocimiento desde lo instrumental hasta lo conceptual y formal sobre las estructuras de datos, las nociones de algoritmo y programa, así como de autómata ejecutor (Salanci, 2015; da Rosa et al., 2020).

Es notorio ver la manera en que los autores al describir esta estrategia con ejemplos usando un lenguaje de programación funcional, no hacen otra cosa más que introducir el pensamiento computacional (donde el estudiante enseña al computador a resolver un problema general, es decir programa) desde el pensamiento algorítmico (donde el estudiante resuelve un caso concreto del problema general y diseña una solución algorítmica) (ver da Rosa et al., 2020).

En otras palabras, la didáctica de la programación es la base del modelo didáctico del PC y las ciencias computacionales. En este modelo, al que llamamos modelo STEM(p), la EI cumple un rol preponderante ya que, por un lado, propone utilizar la programación como espacio de reflexión y consolidación de las bases del pensamiento computacional y, por otro lado responde a la pregunta de cómo lograr que los estudiantes experimenten la ciencia de la computación (y la programación) para desempeñarse en las profesiones en las que deberán aplicar las herramientas computacionales en sus niveles más altos de diseño, ya sea en CS o en cualquier otra área. Ante los desafíos del PCC, el impacto de la edTech en las políticas educativas, y la introducción del PC desconectado de su disciplina, reafirmamos la necesidad de retornar al problema de fondo que continúa pendiente: la priorización y desarrollo de la didáctica de CS y su formación (en todos los niveles), y su reconocimiento como ciencia en la educación básica.

## Referencias

Aikat, J., Carsey, T., Fecho, K., Jeffay, K., Krishnamurthy, A., Mucha, P. y Ahalt, S. C. (2017). Scientific training in the era of big data: A new pedagogy for graduate education. *Big Data*, *5(1)*, 12–18. https://doi.org/10.1089/big.2016.0014.

Armoni, M. (2016). Computer science, computational thinking, programming, coding: The anomalies of transitivity in k–12 computer science education. *ACM Inroads, 7 (4)*, 24– 27.

Atkin, J. M. y Black, P. (2005). *Changing the subject: innovations in science, maths and techno- logy education.* Routledge.

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.

Berry, G., Dowek, G., Abiteboul, S., Archambault, J., Balagué, C. Baron, G. y Viéille, T. (2013). *L'enseignement de l'informatique en france, il est urgent de ne plus attendre.* https://www.academie-sciences.fr/fr/ Rapports-

ouvrages-avis-et- recommandations -de-l-Academie/l-enseignement-de-l-informatique-en-france-il-est- urgent-de-ne-plus-attendre.html. (Rapport de l'Académie des Sciences).

Bibri, S. (2021). The core academic and scientific disciplines underlying data-driven smart sustainable urbanism: an interdisciplinary and transdisciplinary framework. *Computational Urban Science, 1(1)*, 1–32.

Blackley, S. y Howell, J. (2015). A stem narrative: 15 years in the making. *Australian Journal of Teacher Education, 40(7)*, 102–112.

Camp, T., W.R., A., Bizot, B., Davidson, S., Hall, M., Hambrusch, S. y Zweben, S. 2017).

Generation CS: the growth of computer science. *ACM Inroads*, *8(2)*, 44–50. https://doi.org/10.1145/3084362.

Cansu, S. K. y Cansu, F. K. (2019). An overview of computational thinking. *Journal of Computer Science Education in Schools*, *3(1)*.

Castells, M. (2000). *The rise of the network society*. Blackwell: Oxford

CECE. (2017). The Committee on European Computing Education (CECE) y Association for Computing Machinery (ACM) *Informatics Education in Europe: Are We All En The Same Boat?* New York NY United States, https://doi.org/10.1145/3106077.

Clauss, A. D. y Nelsen, S. F. (2019). Integrating computational molecular mode ling into the undergraduate organic chemistry curriculum. *Journal of Chemical Education*, 8*(2)*, 44–50.

Cone, L., Brøgger, K., Berghmans, M., Decuypere, M., Förschler, A., Grimaldi, E. y Vanermen, L. (2021). Pandemic acceleration: Covid-19 and the emergency digita lization of european education. *European Educational Research Journal, September*. https://doi.org/10.1177/14749041211041793.

da Rosa, S. (2004). Designing algorithms in high school mathematics. *In Lecture Notes in Computer Science vol. 3294. Springer-Verlag*.

da Rosa, S. (2018). Piaget and computational thinking. *CSERC'18: Proceedings of the 7th Computer Science Education Research Conference*, 44–50. https://doi.org/10.1145/3289406.3289412.

da Rosa, S., Viera, M. y García-Garland, J. (2020). A case of teaching practice founded on a theoretical model. *Lecture Notes in Computer Science 12518 from proceedings of the International Conference on Informatics in School: Situation, Evaluation, Problems*, 146–157. https://doi.org/10.1007/978-3-030-63212-0.

Decuypere, M., Grimaldi, E. y Landri, P. (2021). *Introduction: Critical studies of digital education platforms*. Critical Studies in Education, 62:1. https://doi:10.1080/17508487.2020.1866050

Denning, P. (2017). Computational thinking in science. *Sigma Xi, The Scientific Research Society, American Scientist*, *105*. www.americanscientist.org.

Denning, P., Comer, D., Gries, D., Mulder, M., Tucker, A., Turner, A. y Young, P. (1989). Computing as a discipline. *Communications of the ACM, 32(1)*, 9–23.

Denning, P. y Tedre, M. (2015). *Shifting identities in computing: From a useful tool to a new method and theory of science*. En Hannes Werthner and Frank van Harmelen, Eds. Informatics in the Future, Proceedings of the 11th European Computer Science Summit.

Denning, P. y Tedre, M. (2016). The long quest for computational thinking. *Proceedings of the 16th Koli Calling Conference on Computing Education Research*, 120–129.

Denning, P. y Tedre, M. (2019). *Computational thinking*. Cambridge, MA: The MIT Press. Denning, P., y Tedre, M. (2021). Computational thinking: A disciplinary perspective. *Informatics in Education*, *20(3)*, 361–390. https://doi:10.15388/infedu.2021.21

Dowek, G. (2005). Quelle informatique enseigner au lycée? *Bulletin de l'APMEP, nr.480*.https://www.apmep.fr/IMG/pdf/AAA09014.pdf.

Gomes, A. (2007). Learning to program - difficulties and solutions. *Proceedings of the International Conference on Engineering Education ICEE*, 283–287.

Götschi, I., T. and Sanders y Galpin, V. (2003). Mental models of recursion. *Proceedings of the 34th SIGCSE technical symposium on Computer Science Education*, 346–350.

Hazra, R., Singh, M., Goyal, P., Adhikari, B. y Mukherjee, A. (2019). *The rise and rise of inter- disciplinary research: Understanding the interaction dynamics of three major fields– physics, mathematics and computer science.*

Henderson, M. y Romeo, G. (2015). *Teaching and digital technologies: Big issues and critical questions*. Cambridge University Press.

Holmboe, C., McIver, L. y Carlisle, E. (2001). Research agenda for computer science education. En *G. Kadoda (Ed). Proceedings of XIII Psychology of Programming Interest Group*, 207–223.

Hubwieser, P., Armoni, M., Giannakos, M. y Mittermeir, R. (2014). Perspectives and visions of computer science education in primary and secondary (k-12) schools. *ACM Transactions on Computing Education (TOCE), Special Issue on Computing Education in (K-12) Schools.*, *14(7)*.

*Informatics for all: Educating people for the digital age.* (2018). https://www.informaticsforall.org/wp-content/uploads/2020/07/Informatics -for-All- position-paper.pdf.

Komljenovic, J. (2021). The rise of education rentiers: digital platforms, digital data and rents. *Learning, Media and Technology 46(3),* 1–13.

Kong, S. y Abelson, H. (2019). Computational thinking education. *Springer Nature, p. 382*. Koschmann, T. (1996). *Cscl: Theory and practice of an emerging paradigm*. Lawrence Erlbaum.

Lister, R. (2011). Concrete and other neo-piagetian forms of reasoning in the novice programmer. *Proceedings of ACE 2011: The 13th Australasian Computing Education Conference.*, *114*, 9–18.

Lodi, M. y Martini, S. (2021). Computational thinking, between Papert and Wing. *Science & Education volume, 30*, 883—908. https://doi.org/10.1007/s11191-021-00202-5.

Mannila, M. L. and Peltomäki y Salakoski, T. (2007). What about a simple language? analyzing the difficulties in learning to program. *Proceedings of the International Conference on Engineering Education ICEE*, 211–227.



Markowetz, F. (2017). All biology is computational biology. https://doi.org/10.1371/journal. pbio.2002050.

Mayer-Schönberger, V. y Cukier, K. (2013). *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcour.

Mittal, S., Durak, U. y Ören, T. (2017). *Guide to simulation-based disciplines: Advancing our computational future.*

Nicholls, A. (1983). *Managing educational innovations.* https://doi.org/10.4324/9781351040860.

Nwana, H. (1997). The computer science education crisis: fact or illusion? *Interacting with Computers*, 9(1), 27–45. https://doi.org/10.1016/S0953-5438(97)00005-2.

Palm, T. (2008). *Impact of authenticity on sense making in word problem solving.* (Vol.67).https://doi.org/10.1007/s10649-007-9083-3.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Pasterk, S. y Bollin, A. (2017). *Digital literacy or computer science: Where do information technology related primary education models focus on?* https://doi:10.1109/ICETA.2017.8102517.

Paulson, L. (2017). Computational thinking is not necessarily computational. *Communications of the ACM*, *60*.

Pears, A., Tedre, M., Valtonen, T. y Vartiainen, H. (2021a). *What makes computational thinking so troublesome?* https://www.researchgate.net/publication/ 353953443 What Makes Computational Thinking so Troublesome. License CC BY 4.0. http://doi: 10.13140/RG.2.2.20480.35842

Pears, A., Tedre, M., Valtonen, T. y Vartiainen, H. (2021b). *What makes computational thinking so troublesome?* http://doi: 10.13140/RG.2.2.20480.35842

Pollock, L., Mouza, C., Guidry, K. R. y Pusecker K. (2019). Infusing computational thinking across disciplines: Reflections and lessons learned. En *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE'19), Association for Computing Machinery, CB-252*, 435–441. https://doi.org/10.1145/ 3287324.3287469

Roach, E., K. and Tilley y Mitchell, J. (2018). How authentic does authentic learning have to be? *Higher Education Pedagogies*, *3(1)*, 495–509. http://doi: 10.1080/23752696.2018.1462099

Roberts-Mahoney, A., H. and Means, y Garrison, M. (2016). Netflixing human

capital development: Personalized learning technology and the corporatization of k-12 education. *Journal of Education Policy 31(4)*, 405–420.

Roth, W. M., Van Eijck, M., Reis, G. y Hsu, P. L. (2019). Authentic science revisited: En praise of diversity, heterogeneity, hybridity. *Brill, Leiden*.

Saeli, M., Perrenet, J., Jochems, W. y Zwaneveld, B. (2011). Teaching programming in secondary school: A pedagogical content knowledge perspective. *Informatics in Education, 10(1),* 73–88.

Salanci, L. (2015). Didactics of Programming. *ICTE Journal*, 4(3), 32-39.

Sancho-Gil, J., Rivera-Vargas, P. y Miño-Puigcercós, R. (2020). Moving beyond the predictable failure of ed-tech initiatives. *Learning, Media and Technology 45(1)*, 61– 75.

Sanders, M. (2009). *Stem, stem education, stemmania.* https://vtechworks.lib.vt.edu/bitstream/handle/10919/51616/STEMmania.pdf.

Schwill, A. (1997). Computer science education based on fundamental ideas. *Proceedings of the IFIP TC3 WG3.1/3.5 joint working conference on Information technology: supporting change through teacher education*, 285–291.

Selwyn, N. (2013). *Distrusting educational technology: Critical questions for changing times.* https://doi.org/10.4324/9781315886350.

Serdyukov, P. (2017). Innovation in education: what works, what doesn't, and what to do about it? *Journal of Research in Innovative Teaching and Learning 10(1)*, 4–33.

Sharma, A. (2017). A model scientific computing course for freshman students at liberal arts colleges. *Journal of Computational Science Education*, 2–9.

Sharma, A. y Asirwatham, L. (2019). *Learning by computing: A first year honors chemistry curriculum.*

Tedre, M. (2020). From a black art to a school subject: Computing education's search for status. *ITiCSE '20: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 3–4.

UNESCO, (2020). *Education: From disruption to recovery.* https://en.unesco.org/covid19/educationresponse

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L. y Wilensky, U. (2015). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 127–147.

Williams, P. (2011). Stem education: Proceed with caution. *Design And Technology Education: An International Journal, 16(1)*, 61–75. https://ojs.lboro.ac.uk/DATE/article/view/1590.

Williamson, B. (2017). Learning in the 'platform society': Disassembling an educational data assemblage. *Research in Education 98(1)*, 59–82.

Wilson, C., Sudol, L., Stephenson, C. y Stehlik, M. (2010). *Running on empty: The failure to teach k12 computer science in the digital age.* Association for Computing Machinery, ISBN 9781450388672.

Wing, J. (2006). Computational thinking. *Communications of the ACM*, *49(3)*.

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical transitions of the Royal Society*, *Phil. Trans. R. Soc. A 366*, 3717–3725.

Wing, J. (2010). Computational thinking–what and why? *The Link*. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why