Didáctica computacional: modelado computacional y desarrollo del pensamiento computacional con MateFun

Manuela Cabezas[1] and Sylvia da Rosa[2]

[1] Facultad de Ciencias de la Educación, Universidad de la Empresa,
mcabezas@ude.edu.uy
[2] Instituto de Computación, Facultad de Ingeniería, Universidad de la República
darosa@fing.edu.uy

Resumen En este paper presentamos un analísis de una selección de secuencias didácticas que fueron elaboradas e implementadas en clase por profesores de matemática de enseñanza media, principalmente bachillerato (EMS). Los profesores participaron de un proyecto de investigación titulado "Espacios de investigación didáctica en el área de matemática y computación", liderado por nuestro grupo de investigación en didácticas computacionales y con la colaboración de la Sociedad de Educación Matemática del Uruguay (SEMUR). Partimos del modelo didáctico que presentamos en SAEI 2025 y de una definición de pensamiento computacional que pone el foco en el modelado computacional. Este modelo es especialemte apto para los niveles superiores de la enseñanza media, donde el paradigma computacional introduce la computación como ciencia e impone una revisión de contenidos y problemas nuevos en el curriculum. El proyecto buscó aportar recursos conceptuales y prácticos para el abordaje del pensamiento computacional en aulas de matemáticas. Para ello analizamos las secuencias en cuanto a los aspectos críticos del contenido y dificultades prácticas del modelado computacional de soluciones a problemas de matemática. El análisis de las secuencias didácticas confirma la complejidad que representa introducir cambios que implican una reconceptualización epistemológica de los contenidos. En este artículo presentamos ejemplos seleccionados que ilustran la situación y discutimos los resultados. Comprobamos que los profesores tuvieron dificultades en aplicar los conceptos de las etapas iniciales del modelo y que la reformulación de los enunciados como problemas algorítmicos está ligada a la falta de experiencia y estrategias para detectar aspectos computacionales. Finalmente se discute el significado de las dificultades encontradas y se delinean algunas actividades a futuro.

Palabras clave: didáctica, modelado, programación y matemática

1. Introducción

El "Pensamiento Computacional" se introduce como competencia transversal en el marco curricular nacional (MCN) con la reforma educativa que se imple-

menta a partir de marzo 2023 en Uruguay (ANEP, 2022, p.44). El pensamiento computacional implica un área relativamente nueva en el curriculum y con poca experiencia comprobada en Uruguay (García, 2020). Sin embargo, es conocida desde otros países la dificultad de introducción del pensamiento computacional en el campo de la educación y la complejidad que esto implica para la introducción en la educación básica (Bocconi y col., 2022);(Cansu y Cansu, 2019). En Uruguay, si pensamos en la integración del pensamiento computacional en la formación docente, el trabajo es dirigido principalmente por Ceibal y para el nivel de primaria (García, 2020); (INEEd, 2024, p.26). En parte entendemos que esto se debe al conocimiento más generalista de los maestros en comparación con niveles superiores donde los contenidos específicos tienen mayor peso y las aplicaciones concretas deben desarrollarse de forma específica. Sin embargo, en otros países, son justamente las asignaturas de matemáticas y ciencias las que presentan un mayor grado de integración del pensamiento computacional en la formación docente (Sundtjønn y col., 2024) ya que es justamente en éstas áreas que surge el pensamiento computacional como producto de la aplicación de conceptos y técnicas computacionales fuera de las ciencias de la computación (Denning y Tedre, 2015, 2016, 2021).

Nuestro grupo de investigación viene desarrollando desde 2020 una línea de investigación en didácticas de las ciencias computacionales que consiste en llevar adelante estudios teóricos y empíricos con el objetivo de elaborar herramientas, teóricas y prácticas, que aporten soluciones a los desafíos que enfrentan los profesores, especialmente en disciplinas científicas, al introducir el pensamiento computacional en sus clases. En años y proyectos recientes hemos ido construyendo conocimiento y modelos, partimos de los fundamentos epistemológicos de la construcción de conocimiento sobre estructuras de datos y programas (da Rosa y Gómez, 2022), identificamos ideas fundamentales y aspectos críticos y formulamos un modelo didáctico para ciencias computacionales (Cabezas y da Rosa, 2025)[3]. El último proyecto realizado durante 2024 y 2025, se tituló "Espacios de investigación didáctica en el área de matemática y computación" [4] y fue dirigido por nuestro grupo de investigación en didácticas computacionales[5].

En este paper presentamos algunos resultados del proyecto, en particular, sobre la etapa de diseño de secuencias didácticas que elaboraron los profesores que participaron en el proyecto. Analizamos los productos que salieron de esta etapa para entender cuándo y de qué forma los profesores lograron (o no) identificar los aspectos computacionales para el contenido/tema que seleccionaron. Nuestro principal resultado, cuyo argumento desarrollamos en este paper, se refiere a las dificultades en la formulación del problema algorítmico, que es representado por la primera línea de nuestro modelo didáctico (ver Sección 3).

El texto se organiza de la siguiente manera: en la Sección 2 se describen los lineamientos metodológicos usados en el proyecto de investigación, en la Sección 3

[3] En adelante, todas las referencias al modelo didáctico corresponden a esta cita.

[4] El proyecto contó con la colaboración de la Sociedad de Educación Matemática del Uruguay (http://semur.edu.uy/sitio/) como institución contraparte.

[5] https://didacticascomputacionales.uy/

se introduce la definición de pensamiento computacional, diferenciándola de la de pensamiento algorítmico, donde la importancia de la construcción de programas y su ejecución radica en los aspectos críticos que surgen en esa etapa; en la Sección 4 se presentan tres ejemplos de secuencias diácticas elaboradas por los profesores. Finalmente en la Sección 5 se incluyen reflexiones sobre debilidades de nuestro modelo didáctico y se esbozan líneas de trabajo futuro.

2. Metodología

Metodológicamente compartimos la visión de quienes reclaman desde hace ya muchos años que es necesario contar con formas innovadoras de vincular las comunidades académicas a la práctica educativa (Hartell y col., 2021; Lunde y Sjöström, 2021; Nutley y Jung, 2008). En este proyecto implementamos "investigación cercana a la práctica" (Parsons, 2021) que busca producir resultados y productos concretos y de utilidad para la práctica. Las metodologías basadas en este modelo posicionan al profesional docente como agente en la construcción de las evidencias que constituyen la base científica de sus prácticas (Lunde y Sjöström, 2021; Runesson, 2011). Esto implica también partir de la práctica en colaboración con una contraparte (SEMUR) e integrar el desarrollo profesional docente en la metodología. Los componentes de la investigación están ligados tanto al desarrollo de conocimiento científico como a la propia práctica docente en términos de objetivos, método, forma de trabajo, análisis y difusión (Serder y Malmström, 2020).

El objetivo general fue generar vínculos por medio de la "investigación cercana a la práctica" sobre la introducción del pensamiento computacional en aulas de matemáticas. A su vez, esta vinculación con la práctica da continuidad al desarrollo del modelo didáctico en sus fases de aplicación, testeo y evaluación (Cabezas y da Rosa, 2025).

Siguiendo autores del campo educativo (Hirschkorn y Geelan, 2008; Levin, 2013) se abordaron los siguientes aspectos:

1. el desconocimiento de la investigación disponible en el área y la desvinculación entre investigadores de área y los beneficiarios de la investigación;
2. la inadecuada formulación de problemas y preguntas de investigación didáctica y el desarrollo de actividades académicas alejadas de, e inaplicables en, la realidad didáctica que aborda el pensamiento computacional;
3. la falta de conocimiento, reconocimiento y espacios de producción colaborativa entre las áreas de dominio (investigacíon didáctica-docencia en educación media);

Estos factores impactan directamente en el desarrollo profesional docente pero también en el campo de conocimiento, que requiere cercanía al campo de práctica como "agente esencial para la correcta formulación/análisis de preguntas y enfoques en el modelado didáctico" (Cabezas, 2023, p. 50).

El diseño incluyó dos grandes estapas, sencuenciadas para introducir a los profesores al campo de conocimiento con una Revisión Sistemática de Literatura

(RSL) y luego trabajar desde el dominio de práctica profesional para producir material didáctico sobre contenidos específicos de matemáticas que aporten directamente al trabajo de aula.

2.1. Actividades y productos

Durante el período de junio a octubre de 2025 se realizó la RSL con la plataforma web y móvil Rayyan que permite trabajo colaborativo con revisión ciega. Se utilizó el protocolo NIRO-SR (Topor y col., 2023) y se organizó un grupo de discusión con todo el equipo para definir criterios de inclusión y exclusión y para definir categorías de análisis de las evidencias seleccionadas. Partimos de la definición de pensamiento computacional del grupo de investigación (Cabezas y da Rosa, 2025) que pone el foco en la tarea de implementar algoritmos y ejecutar los programas. Los encuentros se grabaron por ZOOM y se compartieron en google drive permitiendo repaso y revisión en todas las etapas del proyecto. El análisis abordó la conceptualización y operacionalización del pensamiento computacional y las fortalezas y debilidades de las evidencias encontradas (total 24). El proceso y resultados de la RSL serán presentados en trabajo futuro.

Posterior a la etapa de la RSL (octubre 2025-febrero 2026), se llevaron adelante encuentros de trabajo sincónicos y trabajo asincónico para el diseño de secuencias didácticas. También se trabajó en forma individual con cada profesor en la etapa final para profundizar la discusión del contenido específico que decidió abordar. Las secuencias fueron ajustadas en forma iterativa, discutiendo en grupo e individualmente los aspectos computacionales que fueron surgiendo y ajustando las versiones de cada uno (ver Sección 4). Los profesores, a su vez, pudieron aplicar sus secuencias con sus estudiantes antes de hacer sus reflexiones finales y subir su trabajo en el repositorio que se encuentra en la página del grupo de investigación (https://didacticascomputacionales.uy/). Algunos profesores también presentaron sus trabajos en eventos académicos llevados a cabo en febrero de 2026: 1) en la IV Conferencia y Escuela de Verano sobre Investigación en Educación STEAM en Uruguay los días 5 y 6 y en Argentina los días 11, 12 y 13 en conjunto con la Universidad Nacional de Avellaneda - Sede Piñeyro y 2) en cursos de verano organizados por el Instituo de Profesores Artigas, en donde dirigieron talleres a estudiantes de formación docente en matemática, los días 24 y 26.

3. Marco teórico: pensamiento computacional, pensamiento algorítmico y aspectos computacionales

Antes de introducir el análisis de las secuencias didácticos es necesario hacer algunas aclaraciones sobre nuestra base conceptual en cuanto a la expresión "pensamiento computacional". Cuando hablamos de aspectos computacionales es importante marcar un diferencial con el tipo de definiciones de PC que vemos comunmente, como la de Gambrell, accesible en (Gambrell, 2024)[6]:

[6] La separación en puntos 1 y 2 es nuestra.

"es una forma especializada de pensamiento que involucra

1. la descomposición de problemas, la identificación o reconocimiento de patrones, la abstracción de sistemas complejos junto con la justificación apropiada, y la construcción de algoritmos orientados a resolver problemas que requieren modelado numérico
2. (cuya resolución suele involucrar el uso de computadoras)."

Observar que el foco está puesto en el ítem 1, mientras que lo que figura en el ítem 2 está entre paréntesis, sugiriendo que es secundario. Estamos de acuerdo con lo que señala en el ítem 1 son técnicas matemáticas que pueden utilizarse sin necesidad de programar, de hecho, se han empleado y se emplean en la resolución de problemas en matemáticas y en ciencias desde sus inicios. Es decir, son elementos inherentes a las construcción cientifica de conocimiento y acompañan la evolución de las ciencias, siglos antes del uso de las computadoras. La consecuencia del uso de dichas técnicas es el desarrollo *del pensamiento algorítmico.*

Sin embargo, para desarrollar el pensamiento computacional las técnicas señaladas en el ítem 1 no son suficientes, *se necesita construir el programa, ejecutarlo, analizar el resultado de la ejecución y corregir eventuales errores*, como se explicita en las etapas de nuestro modelo didáctico. En este proceso surgen *aspectos nuevos* (emergentes) que no aparecen en el item 1) y que son los que hacen que se opere *computacionalmente.* La consecuencia de dichas operaciones, aplicando técnicas de programación, es el desarrollo *del pensamiento computacional.*

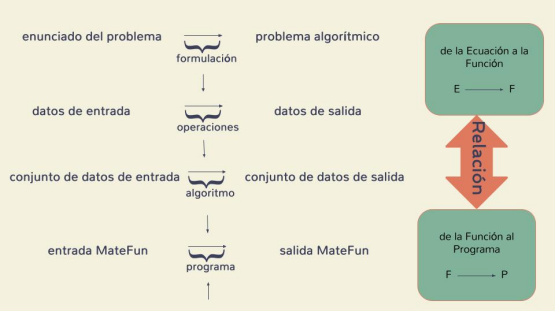
¿Por qué se relaciona a las técnicas del pensamiento algorítmico con el pensamiento computacional? Los orígenes de este concepto erróneo se deben a que recién con la llegada de la computadora se toma conciencia del uso de las técnicas que desarrollan el pensamiento algorítmico y el mismo se generalizó *cuando se introdujo la solución computacional de los problemas.*

Compartimos la idea con otros autores que es un error considerar modelos o definiciones desvinculadas de la programación como propuestas para el desarrollo del pensamiento computacional (Cabezas, 2021; da Rosa, 2018).

Nuestro modelo didáctico fue diseñado como mapa conceptual de los componentes que hacen posible la modelación de soluciones generales (computacionales), como se muestra en la Figura 1. Nuestra definición de pensamiento computacional pone foco en el modelado computacional para programar soluciones donde se evidencian (emergen) aspectos computacionales críticos.

La definición de aspectos críticos que permite conceptualizar los aspectos computacionales dentro del modelo didáctico proviene de la teoría de la variación de Ference Marton (Marton, 2015) que indica que el aprendizaje sucede en funcion del discernimiento (de aspectos críticos), lo cual requiere experimentar variación de los aspectos críticos y no críticos del objeto de aprendizaje. Los aspectos críticos "... can in no way be chosen, or decided upon, as educational objectives. Critical aspects and features have to be searched for and found."(Marton, 2015, p. 26). Para encontrar y *discernir los aspectos críticos del objeto* se desarrollaron las actividades y el análisis de este estudio.

Figura 1: Versión actual del modelo didáctico



En la próxima sección describimos algunas de las secuencias didácticas elaboradas por los profesores que participaron en el proyecto, analizando los aspectos computacionales que quedan en evidencia durante el proceso de modelado computacional y mostrando cómo son (o no son) identificados por los profesores.

4. Secuencias didácticas

Se describen secuencias didácticas sobre los siguientes temas: divisibilidad, modelizar una situación y función cuadrática y se interpretan en una breve síntesis las posibles causas del apartamiento entre lo planificado por los profesores y lo esperado por nuestor equipo. Analizando críticamente las razones de dicho apartamiento, revisamos el proceso por el cual trabajamos el concepto de pensamiento computacional durante el proyecto con los profesores.

4.1. Divisibilidad

En el penúltimo año del bachillerato (16 años de edad) se introduce el tema divisibilidad. Una de las profesoras participante del proyecto, decidió trabajar este tema con sus estudiantes, planteando entre otras, la siguiente actividad, basándose en nuestro modelo didáctico:

1. Dados dos naturales m y n, con $n \neq 0$, ¿Cómo podemos saber si m es múltiplo de n (o si n divide a m)?
2. Diseña un posible algoritmo que permita determinar para cualquier par de números pertenecientes al conjunto de los números naturales si uno es múltiplo del otro.
3. Implementa el algoritmo diseñado en el paso anterior en MateFun, definiendo conjuntos y funciones necesarias.
4. Ejecutar y probar el programa realizado. Verificar errores. Probar con ejemplos.

Para ello, introdujo las definiciones tradicionales de múltiplo y de número divisible por otro (donde las variables representan números naturales) que se muestran

Figura 2: Definiciones tradicionales

- Definición de múltiplo: $m = n \Leftrightarrow \exists\, a \in \mathbb{N} / m = n \,.\, a$
- Desde la relación de que si $m = n \rightarrow n \mid m$

en la Figura 2. El primer renglón se lee: m es múltiplo de n si y solo si existe a tal que $m = n \times a$ (¡Observar el múltiple uso de punto!) y el segundo: si m es múltiplo de n, entonces n divide a m.

Las definiciones son útiles para responder el primer ítem de la actividad para casos concretos. Por ejemplo para saber si 12 es múltiplo de 3, se divide 12 por 3 y se ve que existe 4 que cumple a condición de la definición. Para responder ítem 1 para el caso general, es necesario resolver el resto de los ítems de la actividad: diseñar un algoritmo (ítem 2), construir un programa (ítem 3) y ejecutarlo para cualquier par de números (el segundo distinto de 0) (ítem 4). La profesora observó que en MateFun (ver documentación del lenguaje en el sitio mencionado en nota al pie de página 2) existe la función **rango**, que dados tres números naturales a, b, c con $a < b$ devuelve la secuencia de naturales $a, a + c, a + 2 * c, a + 3 * c$, etc, e incluyó en su secuencia didáctica la siguiente pregunta *Dados dos naturales a y b ¿cómo podemos encontrar todos los múltiplos de a que no superen el valor de b?*. La solución de la profesora en MateFun y un caso de uso se muestra en la Figura 3.

Figura 3: Función múltiplos de un número usando la función rango.

```
1 {- definicion de multiplos de a hasta c -}
2 multiplos :: Nno0 X Nno0 -> N*
3 multiplos (a,c) = rango(a,c,a)
4 {- Ejemplo
5 >multiplos(3,20)
6 [3,6,9,12,15,18]
7 -}
```

La estrategia de la profesora para resolver el resto de los ítems consiste en usar su función *múltiplos* para generar la secuencia de múltiplos de m y comprobar si n se encuentra en ella, lo cual es una solución algorítmica y computacional válida. Sin embargo, implica definir y programar algunas funciones adicionales (por ejemplo la de pertenencia de un elemento a una secuencia), lo que excedería el objetivo de introducir los conceptos básicos de divisibilidad. Finalmente la profesora trabajó en clase solamente con la pregunta adicional y el programa de la función múltiplos usando rango. ¿Cómo podría haberse encarado este ejercicio? Observar que en el contenido de divisibilidad se encuentra la definición de

división entera que es la siguiente: *Dados dos naturales a y b con b distinto de cero, dividir a por b consiste en encontrar dos números naturales q y r llamados cociente y resto respectivamente, que verifican:*

1. $a = bq + r$
2. $r < b$

Los números a y b se denominan dividendo y divisor respectivamente.

A partir de la definición de división entera, se puede definir múltiplo de un número natural usando la función mod que determina el resto de la división entera:

Dados dos naturales m y n con $n \neq 0$, decimos que m es múltiplo de n si el resto de la división entera entre m y n es 0.

Esta es una definición que puede usarse para diseñar el algoritmo del ítem 2 y programarse inmediatamente (ítem 3), ya que en (casi) todo lenguaje de programación, las funciones para hallar el resto r y el cociente q de la división entera están predefinidas o pueden definirse fácilmente, como puede verse en la Figura 4[7].

Figura 4: Función múltiplo usando la definición de división entera.

```
{- Determinar si un numero es multiplo de otro usando la
 funcion mod para hallar el resto de la division entera.
 -}
amultb :: N X Nno0 -> Bool
amultb (a,b) = True si mod(a,b) == 0
             { False
{- Ejemplo
>amultb(12,3)
True
>amultb(0,3)
True
>amultb(17,3)
False
-}
```

Síntesis interpretativa

Dado que la profesora planteó las actividades a trabajar considerando las etapas de nuestro modelo didáctico (los ítems 1, 2, 3 y 4 de la actividad), nos preguntamos qué fue lo que impidió que lograra diseñar y poner en práctica una secuencia didáctica para resolverlas. Hay una razón de fondo que está en la relación dialéctica entre el programa y el algoritmo: observar que al resolver

[7] Observar que esta solución incluye al 0, que no figura en la lista generada por la función múltiplos.

el caso concreto de determinar si 12 es múltiplo de 3, se divide 12 entre 3 y se ve que existe 4 que cumple la definición tradicional *¡pero desde el punto de vista computacional lo importante es que el resto de esa división es 0!* O sea, el aspecto computacional crítico que está implícito en la solución matemática del caso concreto (el uso de las funciones **mod** y **div**), se vuelve explícito al construir la solución computacional general. Vemos una analogía con lo mencionado en la Sección 3 sobre un aspecto de la historia de las ciencias: las técnicas que caracterizan el pensamiento algorítmico se volvieron conscientes, cuando hubo que construir soluciones computacionales. Es decir, los aspectos computacionales que es necesario tener en cuenta influyen en la estrategia con la cual se diseña el algoritmo, de ahí el carácter dialéctico de la relación algoritmo-programa.

El hecho de que los profesores se atengan a los contenidos tradicionales revela un comportamiento característico del paradigma pre-computacional de las ciencias (en este caso de la matemática), donde la computación es una herramienta al servicio de contenidos prefijados a los cuales debe adaptarse. Cabe señalar que el impacto de las soluciones computacionales se extiende a otras partes del contenido, por ejemplo, los criterios de divisibilidad dejan de tener sentido: para saber si un número es divisible por otro, se lo preguntas a tu programa. Esto significa que los criterios de divisibilidad ya no son una forma relevante de averiguar si un número es divisible por otro, sino que pueden ser estudiados como propiedades de los números.

4.2. Modelizar una situación

En la Figura 5 se muestra un ejercicio planteado por un profesor en su secuencia didáctica sobre el tema funciones en el penúltimo año del bachillerato (16 años de edad).

Figura 5: Modelizar

14. El total de una factura telefónica mensual $t(m)$ tiene un costo fijo de \$190 más un adicional de \$25 por minuto m.

    a) Dominio: _____

       Codominio: _____

    b) Completa la tabla:

m	$t(m)$
2	
3	

    c) Escribe una fórmula que modelice la situación:

$$t(m) =$$

Este tipo de enunciado donde se pide una fórmula para modelizar una situación aparece con frecuencia en los ejercicios planteados por los profesores. La

pregunta que surge casi de inmediato en todas los casos es: ¿Por qué pasar de la situación descrita en el enunciado a tener una fórmula que exprese la relación numérica entre los datos de entrada y de salida, sería *modelizar la situación*? El punto es relevante dado que nuestro modelo didáctico aspira a contribuir en el desarrollo de la comptencia de modelado matemático y computacional de los profesores y de los estudiantes.

El modelado matemático y computacional de una situación significa, entre otras cosas, tener en cuenta los tipos de datos representados por las cantidades numéricas, para lo cual (casi) todos los lenguajes de programación proveen de estructuras apropiadas (por ejemplo, record y record case en Pascal, conjuntos en MateFun, etc.). En este caso, el dato de entrada **m** representa minutos y el dato de salida **t(m)** representa pesos, que son casos particulares de conjuntos de datos de entrada y de salida formados por unidades de tiempo y monedas respectivamente. El modelado computacional en MateFun y un ejemplo de uso se muestra en la Figura 6

Figura 6: Conjuntos y funciones que modelizan el costo de una llamada telefónica

```
20 conj Tiempo = { Horas, Minutos, Segundos }
21 conj Moneda = { Pesos, Dolares, Euros }
22 fijo :: () -> (Rno0 X Moneda)
23 fijo () = (190, Pesos)
24 adicional :: () -> (Rno0 X Moneda)
25 adicional () = (25, Pesos)
26 costo :: (Rno0 X Tiempo) -> (Rno0 X Moneda)
27 costo (m) =  (precio, Pesos) si fijo()!2 == Pesos
28           { (precio, Dolares) si fijo()!2 == Dolares
29           { (precio, Euros)
30   donde precio = fijo ()!1 + adicional ()!1 * m!1
31
32 Ejemplo de uso: el resultado de usar el programa para
   calcular el costo de una llamada de 10 minutos es
33 >costo(10, Minutos)
34 (440,Pesos)
```

Síntesis interpretativa

Si bien se puede incluir la mayor parte de la información de los enunciados en los algoritmos matemáticos, ya que los datos se representan mediante conjuntos, estos aspectos críticos (dominio y co dominio de las funciones) que permiten la construcción de soluciones a problemas generales y el verdadero modelado de situaciones, permanecen implícitos e invisibles para los profesores. Al construir la solución computacional y ejecutar el programa se evidencia el potencial didáctico del modelado computacional, que permite aplicar las funciones programadas a numerosos casos de entrada, en este caso por ejemplo a otras monedas y otros

valores del costo fijo y del adicional, simplemente modificando los conjuntos y las funciones constantes,

4.3. Función cuadrática

El caso más notable de la distancia entre la forma tradicional y la forma computacional de encarar los problemas es el de una profesora que planteó en su clase (último año de bachillerato, 17 años) el siguiente problema:

Nos interesa construir una función cuadrática que represente la trayectoria de una pelota en función del tiempo t. Sabemos que en el primer instante la pelota se encuentra en el piso, pasado un minuto la altura es de 1.3 metros y cae al piso en el segundo 7.5. Escribe la función cuadrática que modela la altura de la pelota en función del tiempo, indicando el dominio y el codominio adecuados según el contexto.

La profesora plantea como primer paso de su secuencia didáctica que los estudiantes completen la segunda columna del Cuadro 1, manifestando que su objetivo es que los estudiantes discutan en grupo sobre el enunciado del problema y que aprendan a comunicarse entre sí, haciendo hincapié en la importancia del lenguaje matemático como lenguaje universal para comunicar. Declara que

Cuadro 1: Tabla a completar con los estudiantes

Elemento	Definición
Puntos dados	(0,0), (1,1.3), (7.5,0)
Entrada esperada	valor de tiempo t
Salida esperada	altura h
Expresión analítica	$h(t) = -0{,}2t^2 + 1{,}5t$
Dominio contextual	$t \in [0, 7{,}5]$

espera que los estudiantes resuelvan el problema cambiando la notación o incluso utilizando distintos nombres para la función cuadrática y que el dominio apropiado sea un tema de discusión sobre el cómo se presenta y comunica matemáticamente para que otro lo interprete.

El problema que se plantea en el enunciado, que es hallar la ecuación cuadrática, es secundario en el planteo de la profesora: no se explicita cómo los estudiantes llegarían a completar el renglón "Expresión analítica"de la tabla. Desde la perspectiva de nuestro modelo didáctico es necesario transformar el enunciado en un problema algorítmico: dados los tres puntos, hallar la ecuación cuadrática que pasa por ellos, como se muestra en el Cuadro 2.

La solución computacional se muestra en la Figura 7 donde la función **coefs** devuelve los coeficientes $a \neq 0$, b y c (datos de salida) de una función cuadratica $h : R \to$ R tal que $h(x) = a \times (x - r1) \times (x - r2)$ donde las raices r1 y r2 y el punto (m,n) son los datos de entrada del problema.

Cuadro 2: El problema matemático como problema algorítmico

Datos de entrada	{(0,0), (1,1.3), (7.5.0)}
Datos de salida	$h : R \rightarrow \mathrm{R} \mid h(0) = 0$, $h(1) = 1{,}3$ y $h(7{,}5) = 0$

Figura 7: Función cuadrática

```
35 conj Rno0 = { x en R | x /= 0 }
36 coefs :: R X R X R X R -> Rno0 X R X R
37 coefs (r1,r2,m,n) = (a, -(r1 + r2) * a, r1 * r2 * a)
38     donde a = n / (m^2 - (r1 + r2)*m + r1 * r2)
39
40 conj Tiempo = { t en R | (t >= 0 , t <= 7.5)}
41 h :: Tiempo -> R
42 h (t) = -0.2*t^2 +1.5*t
43 {- Ejecutar para el caso concreto
44 raices 0 y 7.5 y punto (1,1.3)
45 coefs (0, 7.5, 1,1.3)
46 (-0.2000,1.5000,0)
47 -}
```

Síntesis interpretativa

Hemos constatado, en este caso y en otros, una cierta tendencia a priorizar "el valor social de la actividad matemática"por encima de la construcción de conocimiento como un proceso individual. La interacción entre compañeros podría ser fuente de conocimiento si el planteamiento del problema fuera claro desde el inicio, por ejemplo planteando completar la segunda columna del Cuadro 2.

5. Discusión: la construcción del problema algorítmico

Pudimos constatar que el primer paso del modelo, que implica (re)formular el enunciado del problema como problema algoritmico es uno de los grandes desafíos cuando los profesores intentan integrar el pensamiento computacional en sus aulas, en parte porque el algoritmo del cual es necesario partir queda implícito. Los profesores carecen de las estrategias necesarias para volver explícito el algoritmo y de esa forma enfocarse en los aspectos computacionales críticos para pasar de la formulación del problema al diseño de la solución. Un ejemplo claro es el caso de la secuencia para el tema *divisibilidad* donde la profesora plantea correctamente el **qué** hay que hacer (en una actividad de cuatro partes), pero no pudo determinar el **cómo** de una manera accesible a los estudiantes en clase de matemática (usar las funciones **mod** y **div**) (ver Sección 4.1).

Desde la autocrítica, vemos las dificultades de los profesores como una debilidad en el modelo didáctico. El modelo no establace de forma satisfactoria elementos guía que permitan una aplicación práctica partiendo del contexto y

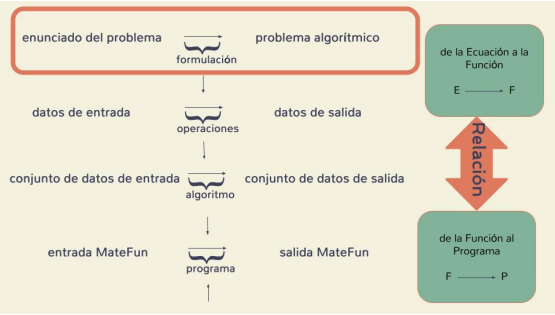
condiciones de cada profesor. Desde el análisis del proceso, constatamos que, como los aspectos computacionales del contenido no son identificados previamente al modelado computacional (descrito en las líneas 2, 3 y 4 del modelo), el modelo en su versión actual lleva a los profesores a cometer errores en la etapa incial (línea 1 del modelo) que impiden una correcta conceptualización del modelado computacional.

Razonamos que si los aspectos computacionales únicamente son identificados cuando se vuelven explícitos en cada caso/ejemplo del modelado computacional, los profesores que no han pasado por múltiples experiencias de modelado no serán capaces de discernir los aspectos críticos en su contenido: un ejemplo de ello es el diseño de la secuencia didáctica sobre la función cuadrática de la Sección 4.3, donde la profesora no explicita el paso fundamental de hallar los coeficientes del polinomio de segundo grado en su diseño inicial.

Esto produce un cuello de botella en la formulación del enunciado como problema algorítmico que requiere varias instancias y correcciones para poder avanzar. En particular, lo referente al pasaje de la ecuación a la función (ver Figura 8) se evidencia en el ejemplo de la factura telefónica en la Sección 4.2, donde el profesor no plantea formular el problema algorítmico sino que el enunciado se centra en la expresión matemática de la ecuación. El problema no es identificado por el profesor, previamente a las etapas de "operaciones", "algoritmo" y "programa"del modelo, indicadas en la Figura 8.

Recién al discernir los aspectos computacionales (durante el modelado) es que vemos la construcción de conocimiento que permite al profesor pasar del enunciado al problema algorítimico. Siendo así, este paso no puede ser el paso inicial, indicándo a la formulación como requisito del modelado computacional, como se muestra en la Figura 8. Entonces, ¿qué debemos considerar para la etapa

Figura 8: Paso inicial del modelo didáctico



de formulación de problema algorítmico? A continuación describimos algunos lineamientos para dejar planteada una posible respuesta a elaborar en trabajos futuros y eventuales ajustes al modelo. Partimos de la definición de problema algoritmico presentada en (Cabezas y da Rosa, 2022), tomada de D. Harel en

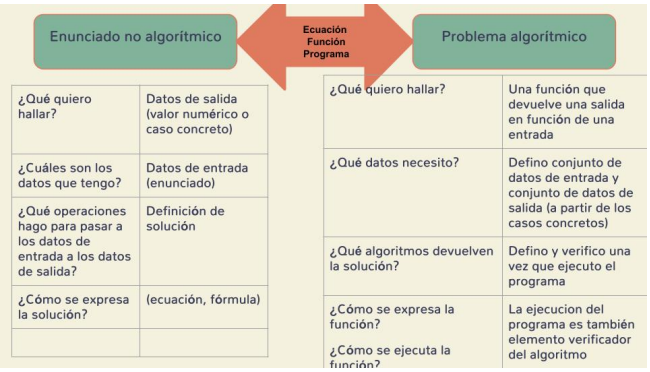
Harel y Feldman, 2004 (p. 16), que plantea que si bien no existe una definición formal de algoritmo, sí la hay para *problema algorítmico* y es la siguiente:

1. una caracterización de una colección legal, posiblemente infinita, de potenciales conjuntos de entrada,
2. una especificación de las salidas deseadas en función de las entradas.

Desde esta definición de problema algorítmico y nuestros resultados buscaremos convertir al proceso de modelado computacional en objeto de aprendizaje para la correcta formulación del problema algorítmico.

En efecto, considerando la definición del aprendizaje en funcion del discernimiento (de aspectos críticos), y que esto requiere experimentar variación de los aspectos críticos y no críticos del objeto de aprendizaje (Marton, 2015), vemos que es requisito que tanto los profesores como sus estudiantes experimenten variación del objeto de aprendizaje (en este caso, el modelado computacional, abarcado por las líneas 2, 3 y 4 del modelo) y así desarrollar el pensamiento computacional. Así queda definido el pensamiento computacional en relación a la capacidad de formular y solucionar problemas algorítmicos (línea 1 del modelo, que es donde detectamos los problemas en este estudio). Las preguntas planteadas en la Figura 9 constituyen una guía para repensar la línea 1 (formulación) del modelo didáctico como generador de múltiples experiencias de modelado.

Figura 9: Etapa a incluir en el modelo didáctico



En conclusión, para dar continuidad a nuestra línea de investigación y enfocar trabajos futuros, es necesario considerar el proceso de modelado en sí mismo como elemento formador por el cual se llega a discernir los aspectos computacionales críticos, experimentando variación, es decir multiples casos diseñados para su identificación y conceptualización, ya sea en contenidos de matemáticas u otras disciplinas. Entendemos que esto se cumple tanto en la toma de consciencia del profesor para su diseño de la secuencia, como en los procesos de aprendizaje de sus estudiantes (para que ellos puedan discernir los mismos aspectos).

Referencias

ANEP. (2022). *Marco Curricular Nacional.* (Reporte de la Transformación Educativa Administración Nacional de Educación Pública).

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M., Jasutė, C., E.and Malagoli, Masiulionytė-Dagienė, V. & Stupurienė, G. (2022). Reviewing Computational Thinking in Compulsory Education. *In Publications Office of the European Union, Inamorato Dos Santos, A., Cachia, R., Giannoutsou, N. and Punie, Y. editor(s),Luxembourg*, *ISBN 978-92-76-47208-7*(doi:10.2760/126955, JRC128347). https://https://doi.org/10.35305/revistairice.vi49.2035

Cabezas, M. (2021). Pensamiento Computacional, Educación STEM y la Educación Informática: Cuestiones Pendientes. *"Revista sudamericana de educación, universidad y sociedad (RSEUS).", 9(1)*, 45-59. https://plataformas.ude.edu.uy/revistas/rifedu/index.php/RSEUS.

Cabezas, M. (2023). Didactic Modeling for Specific Didactics: Computational Science. *"Revista sudamericana de educación, universidad y sociedad (RSEUS).", 11(No. Esp).* http://34.95.139.155/index.php/RSEUS/en/article/view/210.

Cabezas, M. & da Rosa, S. (2022). Modelado Didático para Ideas Fundamentales en Computación. *Proceedings of The 51 SADIO Conference, Simposio Argentino de Educación en Informática (SAEI 2022).* https://revistas.unlp.edu.ar/JAIIO/article/view/18293.

Cabezas, M. & da Rosa, S. (2025). Modelado didáctico con el lenguaje de programación MateFun. *Proceedings of The 54 SADIO Conference, Simposio Argentino de Educación en Informática (SAEI 2025). ISSN 2451-7496.* https://https://revistas.unlp.edu.ar/JAIIO/article/view/19970.

Cansu, S. K. & Cansu, F. K. (2019). An Overview of Computational Thinking. *Journal of Computer Science Education in Schools*, *3(1).*

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. & Gómez, F. (2022). The construction of knowledge about programs. *Proceedings of Psychology of Programming Interest Group (PPIG).* = https://ppig.org/papers/2022-ppig-33rd-gomez/.

Denning, P. & Tedre, M. (2015). *Shifting Identities in Computing: From a Useful Tool to a New Method and Theory of Science.* In Hannes Werthner; Frank van Harmelen, Eds. Informatics in the Future, Proceedings of the 11th European Computer Science Summit.

Denning, P. & Tedre, M. (2016). The Long Quest for Computational Thinking. *Proceedings of the 16th Koli Calling Conference on Computing Education Research*, 120-129.

Denning, P. & Tedre, M. (2021). Computational Thinking: A Disciplinary Perspective. *Informatics in Education.*, *20(3)*, 361-390. https://doi.org/10.15388/infedu.2021.21

Gambrell, J. (2024). Multiple Choice Computational Thinking Assessment for Introductory Physics [Tesis doctoral, Drexel University]. https://researchdiscovery.drexel.edu/esploro/outputs/doctoral/Multiple-Choice-Computational-Thinking-Assessment-for/991021902014004721

García, J. M. 2. (2020). La expansión del Pensamiento Computacional en Uruguay. *Revista Educación a Distancia*, *20*(63). http://dx.doi.org/10.6018/red.410441

Harel, D. & Feldman, Y. (2004). *Algorithmics - The Spirit of Computing*. Addison-Wesley Publishers Limited 1987, 1992, Pearson Education Limited 2004.

Hartell, E., Brugge, R., Boberg, A., Kozma, C. & Pears, A. (2021). Synergy in the Interface Between School and the Academy. *In PATT38 The 38th Pupil's Attitudes Towards Technology Conference.*, 27-28.

Hirschkorn, M. & Geelan, D. (2008). Bridging the research-practice gap: research translation and/or research transformation. *Alberta Journal of Educational Research, 54(1).*

INEEd. (2024). INEEd (2024). Uruguay en el ICILS 2023. https://www.ineed.edu.uy/images/publicaciones/informes-de-uso-de-%20tics-en-la-educacion/Uruguay-en-el-ICILS-2023.pdf

Levin, B. (2013). To know is not enough: Research knowledge and its use. *Review of education, 1(1)*, 2-31.

Lunde, T. & Sjöström, J. (2021). Didaktiska modeller som kärnan i ämnesdidaktik: forskning som eftersträvar en professionsvetenskap för lärare. *ATENA didaktik, 3(1).*

Marton, F. (2015). *Necessary conditions of learning*. Routledge.

Nutley, S. & Jung, I., T. & Walter. (2008). The many forms of research-informed practice: a framework for mapping diversity. *Cambridge Journal of Education, 38:1.*, 53-71.

Parsons, S. (2021). The importance of collaboration for knowledge co-construction in 'close-to- practice'research. *British Educational Research Journal, 47(6)*, 1490-1499.

Runesson, U. (2011). Lärares kunskapsarbete-exemplet learning study. Om undervisning och lärande. *Stiftelsen SAF i samarbete med Lärarförbundet*, 6-17.

Serder, M. & Malmström, M. (2020). Vad talar vi om när vi talar om praktiknära forskning? *Pedagogisk forskning i Sverige, 25(1)*, 106-109.

Sundtjønn, T., Aalbergsjø, S. G., Møller, T. E. & Schrøder, V. (2024). Review on pedagogical practices for computational thinking in teacher education: Characterizing an emerging field. *Nordic Journal of Comparative and International Education (NJCIE)*, *8*(4).

Topor, M., Pickering, J. S., Mendes, A. B., Bishop, D. V., Büttner, F., Elsherif, M. M., Evans, T. R., Henderson, E. L., Kalandadze, T., Nitschke, F. T. y col. (2023). An integrative framework for planning and conducting Non-Intervention, Reproducible, and Open Systematic Reviews (NIRO-SR). *Meta-Psychology, 7.*