Kotsopoulos, D., Floyd, L., Khan, S., Namukasa, I. K., Somanath, S., Weber, J., & Yiu, C. (2017). **A Pedagogical Framework for Computational Thinking**. Digital Experiences in Mathematics Education, 1-18. DOI:10.1007/s40751-017-0031-2.

Resumen: Nuestro objetivo es proponer un **marco de trabajo pedagógico para el pensamiento computacional** (CTPF, por sus siglas en inglés), desarrollado a partir de las teorías del construccionismo y el constructivismo social. CTPF incluye **cuatro experiencias pedagógicas**: (1) unplugged/hacer desenchufado, (2) **tinkering**/hacer lúdico, (3) **making/**hacer creativo y (4) **remixing**/hacer con remezcla. Las experiencias unplugged se centran en actividades implementadas sin el uso de computadoras. Las experiencias de tinkering involucran, principalmente, actividades donde se toman las cosas, se desarman y se busca producir cambios/modificaciones a objetos existentes (hackear). Las experiencias de crear (making) involucra actividades cuyo foco es la construcción de nuevos objetos (prototipo). La actividad de remezcla hace referencia experiencias que involucran la apropiación de objetos o componentes de objetos para usar en otros objetos o para otros propósitos. Los objetos pueden ser digitales, tangibles o incluso conceptuales.

Estas experiencias reflejan distintas experiencias de PC que se solapan, las cuales están se proponen como necesarias para que los estudiantes experimenten de manera completa el PC. En algunos casos, especialmente para los principiantes y dependiendo de los conceptos que se están explorando, un **enfoque secuencia**l de estas experiencias puede ser útil.

Un marco de trabajo pedagógico para el pensamiento computacional.

La tecnología está presente en prácticamente todos los aspectos de la vida diaria, incluida la de los niños. Además de estar integrada en las herramientas cotidianas (por ejemplo, dispositivos de comunicación, hornos de microondas, automóviles, relojes de alarma, etc.), las computadoras y los dispositivos móviles para uso personal y educativo son comunes en los hogares y en las escuelas. En Canadá, el 83% de los hogares tiene acceso a Internet y el 97% de estos hogares reportan tener acceso de alta velocidad (Statistics Canada 2013).

El acceso a Internet a través de dispositivos personales y móviles ha creado una red de conocimiento global ampliamente accesible. Este acceso ha facilitado el intercambio de información que permite que los usuarios, sean a a la vez, consumidores y desarrolladores de conocimiento. Las implicaciones y la influencia potencial de esta dualidad de consumidor/desarrollador no están siendo subestimadas por los creadores de políticas y curriculum. En muchas jurisdicciones de todo el mundo se reconoce cada vez más que para funcionar, resolver problemas, participar en la innovación digital y promover una sociedad que ya está basada en la tecnología, los individuos requieren algún nivel de "pensamiento computacional" (PC) y este requisito es a menudo acumulativo, en el sentido que nuevas tecnologías requieren el entendimiento de las anteriores.

PC es "un enfoque para resolver problemas, diseñar sistemas y comprender el comportamiento humano que se basa en conceptos fundamentales de la computación" (Wing 2006, 2008, p. 3717). El PC también se puede ver como un pensamiento algorítmico utilizando los principios de la informática como marco estructural de orientación y, a veces, metafórico (Shodiev 2013). Hoyles y Noss (2015) definen el PC como la abstracción (ver un problema en diferentes niveles de detalle), el pensamiento algorítmico (la tendencia ver las tareas en términos de pasos más pequeños discretos conectados), la descomposición (resolver un problema implica resolver un conjunto de problemas de más pequeños) y reconocimiento de patrones (ver a un nuevo problema en relación a problemas anteriores que se haya encontrado).

El PC incluye conceptos (por ejemplo ciclo, condiciones, subrutinas) y prácticas (por ejemplo abstracción y evaluación) principalmente de la computación que se comparten con otras disciplinas,

como la ciencia, las matemáticas, las ciencias sociales, la biología, el lenguaje y la ingeniería (Kafai y Burke). 2013; Lye y Koh 2014). Estas disciplinas también son, a la vez, computacionales, ya que la tecnología digital es cada vez es más necesaria en cada disciplina (Weintrop et al. 2016). Más allá de la relevancia obvia del PC para la disciplina computación, los académicos argumentan que el PC necesita ser enseñado en otras áreas de conocimiento y desde el jardín de infantes (Barr y Stephenson 2011; Yadav et al. 2011). Se propone que el PC sea "una poderosa habilidad cognitiva que puede tener un impacto positivo en otras áreas del crecimiento intelectual de los niños" (Horn et al. 2012, p. 380); por lo tanto, según Wing (2006), debe agregarse "a la capacidad analítica de cada niño" (p. 33).

El intenso interés en el PC ha dado lugar a que muchas jurisdicciones de todo el mundo propongan que sea introducido como una materia o como una propuesta interdisciplinaria. Por ejemplo, Inglaterra ha hecho de la enseñanza de la programación una parte del currículum nacional en las escuelas primarias y ha sido obligatorio a partir del primer grado (Berry 2013; Gobierno de Inglaterra 2013). La programación es un ejemplo del PC y posiblemente sea el ejemplo más popular a la fecha. Finlandia introducirá un currículo obligatorio de PC elemental a partir del 2016, y Estonia ha implementado un currículum a partir de primer grado para todos los estudiantes desde 2013 (SITRA 2014). También se observan tendencias similares en algunas partes de Canadá (por ejemplo, el Gobierno de Columbia Británica 2016; Provincia de Nueva Escocia 2015) y los Estados Unidos (por ejemplo, La Casa Blanca 2016).

El enfoque actual del PC se puede ver como un enfoque renovado. Los orígenes históricos de los niños que participan en el PC se remontan a más de treinta años, en el trabajo visionario de Seymour Papert, quien desarrolló el software LOGO para permitir que los niños participen en la programación de computadoras (Papert 1980). Es importante contemplar porque el trabajo pionero de Papert no fue ampliamente adoptado en ese momento (Pierce 2013). Sencillamente, la tecnología ha evolucionado a un nivel en el que hay una dependencia o interacción diaria ineludible para la mayoría de las personas. Como lo propone un artículo en línea en su título, “el futuro será construido por aquellos que saben programar” (SITRA 2014). Además, los lenguajes de programación mucho más simplificados, que utilizan componentes basados en bloques, han hecho que la participación del PC y la programación sean mucho más accesibles. Es esta realidad la que sugiere que el enfoque renovado en la programación será duradero y se incrementará cada vez más en la currícula obligatoria. En consecuencia, no incluir la programación para todos los niños puede socavar potencialmente su participación en la construcción del futuro (Kafai 2015).

Nuestro objetivo para este artículo es proponer un marco pedagógico de PC (CTPF), que está arraigado en el construccionismo (Papert 1980, 1987; Papert y Harel 1991) y las teorías de construcción social (Vygotsky 1978). CTPF, tal como lo concebimos, incluye cuatro experiencias pedagógicas: (1) unplugged, (2) tinkering, (3) making y (4) remixing. El CTPF fue desarrollado por los autores y con el apoyo de otros académicos (ver reconocimientos) en un grupo de trabajo centrado en la pedagogía del PC en un simposio sobre el PC (Namukasa et al. 2015). Durante el trabajo en, las cuatro experiencias fueron identificadas en la literatura disponible y luego se exploraron a través de actividades.

El grupo de trabajo incluyó novatos y expertos (maestros, estudiantes graduados e investigadores) en el PC. Las siguientes preguntas fueron fundamentales para nuestras discusiones y exploraciones: (1) ¿Fue una experiencia apropiada para un novato? (2) ¿Algunas de las experiencias fueron más cognitivas o tecnológicamente exigentes? Y (3) ¿Cómo se relacionan estas experiencias entre sí? El refinamiento del CTPF continuó más allá del simposio a través del diálogo y nuestras propias exploraciones individuales con cada una de las experiencias y con el marco. Para algunos de nosotros, la exploración ocurrió a nivel individual. Para otros, la exploración se realizó en entornos “pre-servicios” (pre-services?) y otros exploraron las experiencias en contextos de desarrollo profesional auténticos. Muchos de nosotros exploramos las experiencias y el marco con los niños en las aulas.

Esta exploración adicional inspiró una revisión en el ordenamiento de las experiencias, una aclaración en el rol del uso secuencial de las experiencias, y una reconsideración del rol de las experiencias unplugged. Por ejemplo, nuestro pensamiento inicial sobre el CTPF consideró a estas experiencias como “fases” que podrían ocurrir de forma secuencial (es decir, primero desconectadas, luego modificando, luego creando y finalmente remezclando). En algunos casos, particularmente para los principiantes y dependiendo de los conceptos que se exploran, este puede ser el caso. Las experiencias desenchufadas, como explicamos brevemente, pueden ser útiles antes de cualquiera de las otras experiencias.

En consecuencia, el CTPF propuesto debe reflejar experiencias de PC distintas pero que se superponen, las cuales proponemos son todas necesarias para que los estudiantes experimenten completamente el PC. Enfatizamos desde el principio que nuestro marco propuesto está destinado a contribuir a las discusiones iniciales sobre PC y pedagogía. La necesidad de datos empíricos sobre la eficacia del marco es necesaria. El CTPF propuesto es una consecuencia de la investigación disponible sobre pedagogía del PC, que se acepta ampliamente desde la infancia. Si bien creemos que el CTPF propuesto tiene un gran potencial, puede haber otras experiencias u otras secuencias de la experiencia que pueden ser más poderosas para el aprendizaje del OC y esto se puede descubrir con más investigación.

Nuestro enfoque disciplinaria utilizado para ejemplificar el CTPF propuesto son las matemáticas debido a las conexiones obvias y numerosas entre matemática y el PC, particularmente en las áreas de resolución de problemas, modelado y análisis (Gadanidis 2015; Sneider et al. 2014). Esto no sugiere que PC solo sea relevante para la educación matemática. El PC también se puede extenderse naturalmente a otros planes de estudio (Horn et al. 2012). Como se señaló anteriormente, algunas jurisdicciones están adoptando un enfoque curricular cruzado debido a la relevancia del PC para otras disciplinas (Kafai y Burke 2013; Lye y Koh 2014). Proponemos que el CTPF puede ser útil para el aprendizaje del PC en los niveles de educación primaria y secundaria, y particularmente útil para los aprendices principiantes, incluidos los maestros en formación (pre-service) y en servicio.

Se han propuesto numerosos marcos para el PC, pero, a nuestro entender, aún no se han establecido marcos que describan la pedagogía. Por ejemplo, Brennan y Resnick (2012) describen tres "dimensiones" del PC que incluyen: conceptos computacionales, prácticas computacionales y perspectivas computacionales. Estas dimensiones capturan el qué, cómo y el por qué del PC sin abordar completamente la enseñanza real del PC. Del mismo modo, Weintrop et al. (2016) proponen una "taxonomía de prácticas" que incluye prácticas de datos, prácticas de modelado y simulación, prácticas de resolución de problemas computacionales y prácticas de pensamiento sistémico. Estas prácticas describen al PC TC en términos de acciones. Resnick et al. (2005) describen los "principios de diseño de tareas" que ayudan en gran medida a pensar por qué una tarea particular del PC cuan adecuada es para un objetivo de aprendizaje u objetivo de enseñanza. Resnick et al. (2009) también introduce las prácticas de tinkering y remezclas, términos que también utilizamos para describir las experiencias pedagógicas. Nuestra visión del tinkering y la remezcla tiene un foco de enseñanza distinto que se centra en el aprendizaje en lugar de solo en la acción.

Perspectivas teóricas

Las cuatro experiencias pedagógicas propuestas están profundamente arraigadas en la teoría del aprendizaje denominado construccionismo, propuesto por primera vez por Papert (1987) y por el concepto “zona de desarrollo próximo” de Vygotsky (1978). La teoría del construccionismo de Papert postula que los estudiantes construyen representaciones internas para dar sentido a su entorno y así desarrollar conocimiento. Los aprendices construyen sobre el conocimiento existente a través del aprendizaje activo centrado en la investigación.

Papert y Harel (1991) dejan en claro que el construccionismo es más complejo que el simple aprender a partir del hacer (learning by making). Papert explica el construccionismo, en su propuesta de la National Science Foundation, de la siguiente manera: “Tomamos una visión del aprendizaje como una reconstrucción en lugar de una transmisión de conocimiento. Luego, extendemos la idea de los materiales manipulables a la idea de que el aprendizaje es más efectivo cuando parte de una actividad que el alumno experimenta como la construcción de un producto significativo “ (Papert 1987). El aprendiz juega un rol activo y principal, más que el maestro, en la construcción de sus conocimientos y esta es la consideración central para los maestros, ya que guían las actividades de instrucción en el aula.

La perspectiva de Papert y Harel (1991) es fundamentalmente multimodal en el sentido de que las representaciones internas se construyen a partir del compromiso con los artefactos en el mundo real y éstos pueden incluir sonido, texto, imágenes, movimiento, etc. También tomamos esta visión amplia de los artefactos en nuestra definición de “objetos” que se utiliza con frecuencia para describir e ilustrar el CTPF propuesto. Tal como se utiliza en este contexto, **el objeto pretende describir algo que es digital, tangible o incluso conceptual**. Por ejemplo, un bloque de programación de computadora es un objeto digital. Un segmento de una estructura o bloques de construcción son ejemplos de objetos tangibles. Una variable o fórmula puede verse como un objeto conceptual. Algunos objetos pueden ser tangibles y digitales (por ejemplo, robots) o digitales y conceptuales (por ejemplo, una fórmula en un programa de computadora). En las siguientes secciones, se proporcionan ejemplos de objetos para ilustrar las cuatro experiencias en el CTPF.

Vygotsky (1978) define la zona de desarrollo proximo (ZDP) como "la distancia entre el nivel de desarrollo real determinado por la resolución de problemas independientes y el nivel de desarrollo potencial determinado a través de la resolución de problemas bajo la guía de un adulto o en colaboración con compañeros más capaces" (p. 86). En consecuencia, **el aprendizaje es visto como algo social, colaborativo e interactivo**. Las cuatro experiencias pedagógicas reflejan parcialmente un camino continuo de desarrollo, o zonas de aprendizaje proximas, por lo que c**ada experiencia pedagógica puede reflejar un nivel de demanda cognitiva cada vez más desafiante** que la experiencia pedagógica anterior para los alumnos. Esta perspectiva de desarrollo no debe interpretarse como una sugerencia de que un nuevo aprendizaje solo puede ocurrir dentro de una experiencia de pensamiento computacional en particular u otra. **Estas experiencias tampoco deben considerarse rígidamente secuenciales; las secuencias de las experiencias pueden variar o incluso ser iterativas.**

Por supuesto, todavía se pueden introducir nuevos conceptos durante cualquier aspecto del CTPF; sin embargo, las tareas reales asociadas con algunas de las experiencias propuestas en el CTPF, tales como hacer y remezclar requieren, en nuestra opinión, un nivel más alto de comprensión fundamental y nivel de habilidad de pensamiento computacional que las experiencias unplugged o de tinkering. Por ejemplo, en la experiencia de hacer (makering), es necesario comprender la programación de la computadora para escribir nuevos programas. Del mismo modo, las habilidades esenciales necesarias para la remezcla probablemente se aprendieron primero a través de tinkering (por ejemplo, aplicando modificaciones simples a los programas de computadora existentes). Esto no quiere decir que no haya personas que participen en la programación de computadoras, por ejemplo, sin ningún conocimiento previo. De hecho, las experiencias se perciben como distintas pero superpuestas (ver figura 1). **Nuestro punto es que desarrollar el pensamiento computacional, enfocando el aprendizaje de manera secuencial a través de estas cuatro experiencias podría ser útil, especialmente para profesores y estudiantes con antecedentes de conocimientos sobre el pensamiento computacional limitados, pero la exposición a las cuatro experiencias es necesaria para experimentar el pensamiento computacional en su totalidad.**

Menos obvio sobre el CTPF propuesto al inicio es el aspecto social de la ZPD de Vygotsky. Para explicar nuestra visión, en cada una de las experiencias que describimos brevemente, **la oportunidad de compromiso con “otros” ocurre con frecuencia como parte de la experiencia** (por ejemplo,

actividades de construcción), es a menudo una característica de la experiencia (ver unplugged), o puede ser implícito en la experiencia. Por ejemplo, **con la remezcla, los estudiantes se apropian (o "hackean") el trabajo de otros y esta es una oportunidad para aprender y desarrollar el conocimiento.** En consecuencia, lo social es un aspecto importante de la CTPF.

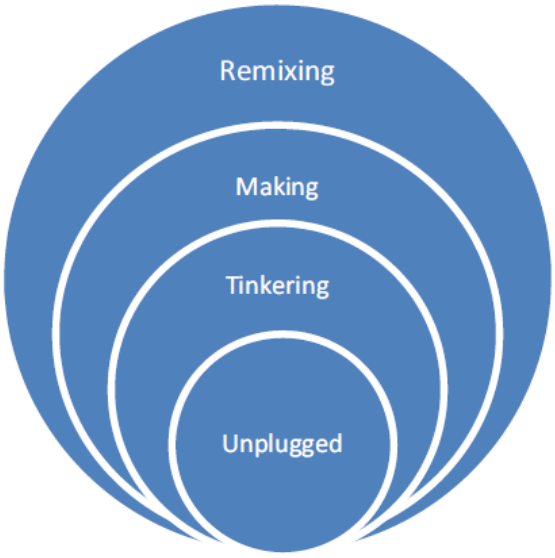

Figura 1 cuatro experiencias pedagógicas

Cuatro experiencias pedagógicas

1- Hacer desenchufado (unplugged)

Las experiencias desenchufadas se centran en las actividades del PC que se implementan sin el uso de computadoras. **Se evitan barreras** tales como el **aprendizaje de un lenguaje** de programación de computadoras o el **acceso limitado a las computadoras**, especialmente para los principiantes y los estudiantes más jóvenes (Curzon 2013; Nishida et al. 2009). En consecuencia, es una forma accesible de introducir a los niños en el PC (Curzon et al. 2014; Thies y Vahrenhold 2013). Estas experiencias liberan "*a los estudiantes de los detalles de implementación necesarios para producir un programa de computadora que funcione*" (Lamagna 2015, p. 52).

Las experiencias desenchufadas a menudo son lo primero y fundamental en el aprendizaje del PC porque requieren posiblemente la menor cantidad de demanda cognitiva y conocimiento técnico. Esto no quiere decir que **las experiencias desconectadas no pueden ser cognitivamente exigentes**. Más bien, **el propósito de las experiencias desenchufadas es introducir conceptos preliminares y superpuestos relacionados con el PC** que luego se pueden explorar de una manera más sofisticada, ya sea conceptual o tecnológicamente, durante las siguiente experiencia. Al igual que las otras tres experiencias, las experiencias desenchufadas también pueden repetirse antes de cualquiera de las otras experiencias para introducir conceptos fundamentales sin el uso de una computadora. Por ejemplo, **un maestro puede hacer una actividad desenchufada antes de crear experiencias para introducir un nuevo concepto relevante sin el uso de la tecnología**.

Las experiencias desenchufadas son a menudo **colaborativas y kinestésicas** (LeMay et al. 2014; Nishida et al. 2009; Taub et al. 2012). Desde una perspectiva de diseño, las experiencias desenchufadas también tienen muchos de los principios de diseño fundamentales descritos por Resnick et al. (2005) que incluyen: múltiples puntos de acceso para los estudiantes, el potencial de desarrollar complejidad dentro de la tarea, son simples y admiten exploraciones.

Con experiencias desenchufadas, los estudiantes pueden presenciar y experimentar el proceso requerido para completar una tarea, lo que les permite ubicar el PC en un contexto relacionado (Curzon et al. 2014). **La incorporación de experiencias desenchufadas no solo ayuda a hacer que los conceptos de ciencias de la computación sean más fáciles de entender, sino que los estudiantes disfrutan trabajar colaborativamente y su motivación e interés en el contenido parecen aumentar** (Curzon et al. 2014; Lamagna 2015). Lambert y Guiffre (2009) **encontraron que las experiencias desenchufadas mejoraron la confianza de los estudiantes de cuarto grado en las matemáticas y su percepción de las habilidades cognitivas**.

Según Nishida et al. (2009), las experiencias desenchufadas se pueden enseñar en cualquier lugar. Sin embargo, si se realiza en una sala con computadoras, los estudiantes pueden ver la actividad desconectada como un preámbulo para trabajar en las computadoras, lo que reduce su atención en la tarea. Curzon (2013) afirma que "las conexiones de las actividades a los conceptos de PC deben ser explícitamente establecidos" (p. 48). **Las experiencias desenchufadas deberían diseñarse para ser dirigidas por los estudiantes, ser kinestésicas, ser fáciles de implementar, estra basadas en juegos y deben incluir desafíos de manera intrínseca** (Nishida et al. 2009). Nishida et al. (2009) también recomienda enseñar las experiencias desconectadas dentro de una única clase, en lugar de trabajarlas varios días.

En lugar de crear nuevas experiencias desenchufadas, los maestros pueden usar las lecciones existentes (es decir, las relacionadas con la clasificación o las que usan los Diagramas de Venn) para incorporar el PC. El desafío al usar el material existente para los maestros es poder identificar el PC en la actividad o poder insertar explícitamente el PC en la tarea donde puede que no haya existido previamente.

Un ejemplo de una actividad desconectada es la clasificación de formas basadas en propiedades y atributos utilizando una estructura simple de árbol de decisiones “si-entonces”. La clasificación es un ejemplo de un algoritmo computacional en el que las propiedades o los atributos forman la regla para colocar un objeto en una colección frente a otras (Namukasa et al. 2015). Esta actividad solo requiere un conocimiento previo limitado de las matemáticas o la programación de computadoras, y puede llevar al desarrollo de conceptos matemáticos y de programación de computadoras más avanzados, tales como el desarrollo de algoritmos de clasificación (Papert 1980; Resnick et al. 2005). Los objetos en un contexto desconectado pueden ser tangibles o conceptuales, en lugar de digitales o basados en computadora. Las experiencias desenchufadas también pueden combinarse con cualquiera de las otras experiencias descritas brevemente.

Otro buen ejemplo de una experiencia desenchufada se encuentra en este volumen especial. Gadanidis et al. (2016) describe una actividad unplugged de lanzamiento de moneda con estudiantes de primer grado que explora el Teorema del binomio (ver artículo y figura 6). En esta actividad, se les pide a los estudiantes que registren y comparen la cantidad de resultados que siguen cada uno de los caminos (1 a 6). La actividad explora probabilidad, estimación y conjetura. Proporciona experiencias concretas y kinestésicas para el niño y permite que se desarrollen conocimientos fundamentales. Refleja Weintrop et al. (2016) que en las prácticas de datos, modelos y prácticas de simulación se incluyen varios de los principios de diseño identificados por Resnick et al. (2005), que incluyen: soporte para muchos niveles de aprendizaje, apoyo en la colaboración y hacen que un concepto matemático sea muy simple y accesible.

2- Hacer lúdico (tinkering)

Las experiencias de hacer lúdico implican principalmente desarmar cosas y participar en cambios y / o modificaciones a objetos existentes. Estos objetos pueden ser bloques de construcción, rompecabezas, simulaciones digitales o electrónicas, código de programación, etc. Durante el hacer

lúdico, los estudiantes no construyen un objeto, digital o de otro tipo, sino que exploran los cambios en los objetos existentes y luego consideran las implicaciones de los cambios.

Estas experiencias pueden requerir que los estudiantes utilicen algunos de los conceptos fundamentales y las habilidades aprendidas durante las experiencias desenchufadas, pero también se pueden introducir nuevos conceptos y habilidades. Al igual que las actividades desenchufadas, las experiencias de hacer lúdico pueden ocurrir antes de cualquiera de las otras experiencias o en secuencia después de las prácticas desenchufadas, si es necesario para los aprendices principiantes.

El objetivo de las experiencias de hacer lúdico es proporcionar un contexto para explorar modificaciones incrementales, sin el desafío adicional cognitivamente exigente de construir realmente el objeto. Aplicación, simulación y resolución de problemas son los focos. Estas experiencias, en última instancia, exploran el "Qué tal sí" (por ejemplo, ¿Qué pasa si cambio esta parte del código? ¿Qué sucede si elimino esta parte de la estructura? ¿Qué sucede si agrego esto al objeto?) Más preguntas (Sneider et al. 2014).

Un buen ejemplo de hacer lúdico es modificar el código de programación existente. Cualquier programación de computadora cuando se ejecuta se manifiesta en última instancia en un sentido físico (por ejemplo, sonido, luces, movimiento, etc.). Este aspecto práctico del hacer lúdico con programas permite a los estudiantes ver fácilmente la conexión entre los cambios en el programa y el resultado, e incluso permite que los estudiantes sepan inmediatamente que ocurrió un error cuando el programa no se ejecuta o corre.

Scratch (Lifelong Kindergarten Group Group en el MIT Media Lab 2016; Resnick et al. 2009; Smith y Neumann 2014; Watters 2011a) es un software de programación visual-gráfico ampliamente utilizado y conocido para niños y novatos. El concepto de "hacer lúdico" sustenta este popular software y lo hace altamente accesible incluso para los estudiantes más jóvenes (Resnick et al. 2009). Los estudiantes usan un método de programación basado en bloques mediante el cual se combinan bloques programados (por ejemplo, ejecutar, repetir, detener, mover, etc.) para construir un código ejecutable. Scratch está diseñado para ser altamente interactivo y accesible: "Simplemente haga click ... y comienza a ejecutar su código inmediatamente. Incluso puede hacer cambios ... mientras se ejecuta, por lo que es fácil experimentar nuevas ideas de forma incremental e iterativa. " (Resnick et al. 2009, p. 63). Si bien los niños pueden usar Scratch para "hacer" durante las experiencias que se describen a continuación, también pueden explorar fácilmente códigos de bloque preexistentes para investigar los conceptos de pensamiento computacional. Smith y Neumann (2014) encontraron que Scratch apoya a los estudiantes al establecer conexiones entre las acciones físicas en la pantalla y los comandos y les ayuda a "razonar y predecir ... acciones, lo que lleva a una comprensión más profunda de los atributos de las transformaciones" (pp. 186-87).

La geometría se explora idealmente a través de software como Scratch durante las experiencias de hacer lúdico. Los maestros pueden seleccionar bloques de codificación prearmados de la extensa biblioteca en línea para permitir a los estudiantes explorar conceptos tales como propiedades, atributos, rotaciones, reflexiones y traslaciones al hacer cambios incrementales o pequeños a los programas existentes. Un ejemplo simple de esto es la modificación de líneas de código para producir diferentes salidas. Por ejemplo, a los estudiantes se les proporciona la secuencia de comandos para hacer un cuadrado y se les pide que realicen modificaciones en la secuencia de comandos para crear un rectángulo (ver imagen 1). El hacer lúdico en esta actividad implica prácticas de modelado y simulación (Weintrop et al. 2016) y apoya la exploración (Resnick et al. 2009; Resnick et al. 2005).

Al jugar con programas prearmados, los estudiantes pueden concentrarse en las líneas de código donde las matemáticas son más evidentes. Los estudiantes pueden concentrarse en las variables y fórmulas matemáticas y, a través de ajustes al código, pueden descubrir inmediatamente lo que harán sus cambios. Al hacerlo, los estudiantes pueden apreciar que "la codificación ofrece nuevas formas de experimentar, representar e investigar los conceptos y las relaciones matemáticas" (Gadanidis 2015,

p. 162). Desde una perspectiva pedagógica, el enfoque en el "qué pasaría si" proporciona un contexto para conjeturar, resolver problemas, generalizar, y predecir: todo lo que puede conducir a una comprensión matemática más profunda. Las oportunidades para facilitar el aprendizaje son extensas durante las experiencias de hacer lúdico.

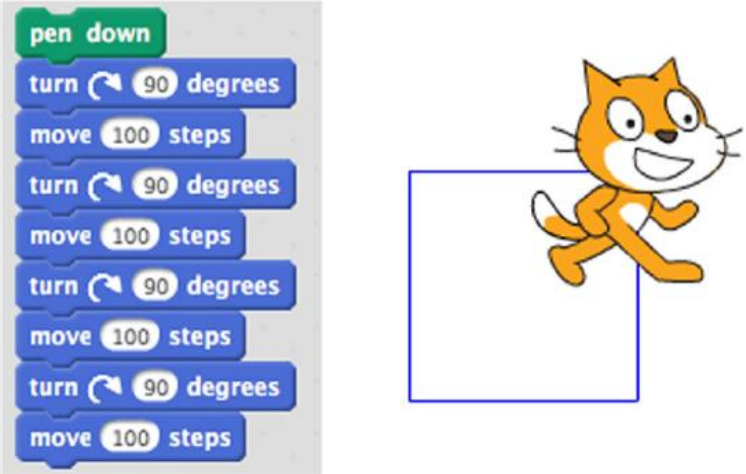

Image 1 Dibujo de un rectángulo con Scratch. Source: http://researchideas.ca/wmt/c6b1.html (Reproduced with permission)

Debemos dejar en claro que nuestra visión de las experiencias de hacer lúdico se alinea con muchas de las prácticas de intercambio definidas como "remix" (remezcla) por Scratch (consulte https://wiki.scratch.mit. Edu / wiki / Remix). Esto es diferente a nuestro "remix" descripto. **La remezcla de Scratch, en su mayor parte, implica compartir y manipular con el código existente y vemos que esto requiere un nivel de demanda cognitiva significativamente menor que nuestra visión de remezclar o incluso hacer.**

3- Hacer creativo (making)

Papert (1980) describió que el aprendizaje consistía en construir un conjunto de materiales y herramientas que uno puede manejar y manipular (handled y manipulate) (p. 173). Las opiniones de Papert son la esencia de las experiencias de hacer creativo. **Estas experiencias son diferentes al hacer lúdico, ya que los objetos se construyen completamente de cero en lugar de modificar objetos preexistentes.** El hacer creativo dependiendo de los objetos utilizados, requiere un conocimiento más profundo que el necesario para el hacer lúdico, donde los objetos en su mayor parte ya están construidos o son preexistentes.

En las experiencias del hacer creativo, se requiere que los estudiantes resuelvan problemas, hagan planes, seleccionen herramientas, reflexionen, se comuniquen y hagan conexiones entre conceptos. A menudo, este hacer involucra prácticas tales como prototipos y pruebas. El conocimiento y el entendimiento que es desarrollado en estas experiencias pedagógicas podrían involucrar el desarrollo de habilidades fundamentales, pero el foco está puesto en usar estas habilidades fundamentales. Los estudiantes tienen el potencial de aprender mientras construyen y mientras comparten lo que hacen, lo que han hecho y cómo lo han hecho (Dougherty 2012). Un "creador" (maker) o "espacio de creadores" (makerspace) son otros términos populares que se refieren a individuos o grupos de personas que construyen cosas y los lugares donde ocurre esto.

El hacer creativo puede ocurrir con cualquier objeto, incluidos aquellos que están explícitamente destinados a la construcción, como Lego o incluso objetos domésticos. En estos casos el hacer está

desconectado. O, el hacer creativo puede ocurrir a través de la programación de computadoras. Estas experiencias también pueden ocurrir al digitalizar objetos o a través de la fabricación digital, a menudo llamada "hacer digital" (Strawhacker and Bers 2015). **La creación digital se refiere a las interacciones entre el mundo real y el virtual a través del uso de interfaces intuitivas que utilizan sensores, microcontroladores, movimiento, sonido, luz, así como otros materiales tangibles para interactuar con el medio ambiente** (Bowler 2014; O'Sullivan and Igoe 2004).
A veces, la creación digital también se conoce como programación tangible (o, computación física, fabricación digital o interfaces de usuario que se pueden agarrar). **La creación digital es simultáneamente una representación tangible y digital del pensamiento computacional que va más allá de la programación y la programación de computadoras basadas en texto.** Los lenguajes de programación orientados a objetos se utilizan a menudo para la creación digital que contrarresta la programación tradicional basada en sintaxis en modo procedural, altamente estructurada y compleja; Por lo tanto, son más accesible para estudiantes (Govender y Grayson 2008). La programación orientada a objetos primero crea objetos a través de la programación donde el caso más abstracto se desarrolla primero y luego se puede modificar o remezclar (ver la sección siguiente) para crear objetos donde las características / comportamientos no esenciales pueden cambiar para obtener variantes. Se proponen lenguajes de programación orientados a objetos para facilitar el aprendizaje de conceptos del pensamiento computacional, como secuenciación, recursión y depuración (Przybylla y Romeike 2014), y para ser útiles en el aprendizaje de las matemáticas (Parker 2012; Scarlatos 2006). El hacer digital anima a los alumnos a combinar múltiples ideas en un proceso cohesivo, organizar su comprensión de nuevas maneras y "depurar" los entendimientos en sus instrucciones para producir algo inesperado" (Wilkerson-Jerde 2014, p. 102). **Ellos son una parte importante de la reaparición de la programación para niños y jóvenes porque tienen el potencial de hacer que los conceptos e ideas abstractos de la programación sean más físicos y accesibles para los niños más pequeños y más susceptibles al aprendizaje social**. El hacer digital también se puede relacionar con las recursos didácticos matemáticos tradicionales, como los bloques, los contadores y las barras de fracciones, que han demostrado apoyar el aprendizaje y la expresión creativa de los niños pequeños (Strawhacker y Bers, 2015, p. 298).

Los tangibles digitales, que son el producto de la fabricación digital,} incluyen, entre otros, textiles digitales (Lovell y Buechley 2011), juegos digitales, bloques programables (Kwon et al. 2012), avatares tangibles/títeres digitales (Liu et al. 2012), materiales reactivos, robótica (Kazakoff et al. 2013; Sullivan et al. 2013), o simplemente algoritmos tangibles. La programación tangible fue el foco clave de las contribuciones seminales de Papert (1980) al campo. A través del LOGO de Papert, los niños aprendieron a programar ambas tortugas, las de pantalla y las del suelo (Papert 1980). Papert se refirió a estas tortugas digitales como "objetos para pensar ... en el que hay una intersección de presencia cultural, conocimiento integrado y la posibilidad de identificación personal" (p.11)

Si bien la mayoría de las investigaciones en esta nueva área han estado orientadas a la innovación, **algunos estudios han investigado la influencia de los tangibles digitales en el aprendizaje**. **Se ha observado que los tangibles digitales son tan efectivos como las interfaces gráficas de usuario para el aprendizaje de lenguajes de programación**, y más útiles en las primeras etapas del aprendizaje de conceptos de programación para niños de kindergarten (Kazakoff et al. 2013; Kwon et al. 2012; Strawhacker and Bers 2015; Sullivan et al. 2013), y para niños con ciertas necesidades especiales (Farr et al. 2010). **Los tangibles digitales también promueven un compromiso más profundo y un enfoque prolongado al explorar conceptos y se ha demostrado que son más atractivos para los niños que solamente programar** (Horn et al. 2012, p. 379).

Se puede relacionar a los tangibles digitales a una mejora de los niveles de interés de los estudiantes de primer año de programación de computadoras (Corral et al. 2014) y a los profesores de ciencias de la computación en servicio y en formación (Govender y Grayson 2008). Bers y Horn (2010) observan que los niños pequeños desde los cuatro años de edad pueden comprender los conceptos básicos de la programación y pueden construir y programar manipuladores robóticos simples a través de

tangibles digitales. Los tangibles digitales también fueron más atractivos para los niños menores de 16 años en el entorno de un museo e igualmente atractivos para niños y niñas en contraste con la programación visual-gráfica que atrajo más a los niños (Horn et al. 2012). En este estudio, sin embargo, los niveles de comprensión de los conceptos de programación obtenidos a través de elementos tangibles e interfaces gráficas no difirieron.

Varios tangibles digitales se pueden adaptar a la enseñanza de conceptos matemáticos como conteo, volumen, área de superficie, ubicación, movimiento, medición, tiempo, distancias, ángulos, etc. El uso de tangibles digitales crea un contexto que es a la vez exploratorio y social. Los estudiantes pueden programar objetos digitales, como robots, bloques digitales o personajes digitales, para representar físicamente conceptos. Sphero (2016), un robot con forma de bola, por ejemplo, que se puede programar para moverse alrededor de un espacio (es decir, aula, mesa, carrera de obstáculos, etc.) utilizando varias aplicaciones de programación de código abierto que utilizan una programación basada en bloque y se puede descargar en dispositivos móviles como tabletas y teléfonos celulares (Imagen 2). El movimiento de programación para Sphero implica pensar en ángulos, longitudes, tiempos, distancias, medidas, así como el razonamiento proporcional. Los estudiantes podrían practicar los conceptos de geometría de transformación programando Sphero para navegar por un laberinto marcado en el piso, de manera similar a navegar por un laberinto en un objeto digital en una pantalla.

Otro ejemplo de hacer digital se puede ver con el uso de la tecnología Arduino (2016). Según Hughes et al. 2016, en este mismo número especial, “Arduino es un kit de desarrollo de circuitos digitales basado en microcontroladores, junto con un entorno de codificación para escribir programas que puedan transferirse al microcontrolador” (pág. 6). Un ejemplo de una actividad de creación que tiene un profundo conocimiento matemático es la programación de patrones en una cuadrícula bicolor utilizando Arduino para encender las luces LED (Hughes et al. 2016; Yiu 2016; ver Imagen 3). La actividad consiste en traducir patrones utilizando el sistema de coordenadas cartesiano. Este sería un ejemplo de pensamiento computacional que involucra prácticas computacionales de resolución de problemas y prácticas de pensamiento sistémico (es decir, incorporar bucles) para hacer que las luces LED se enciende (Weintrop et al. 2016).

Image 2 Sphero coded to complete a path marked on the floor

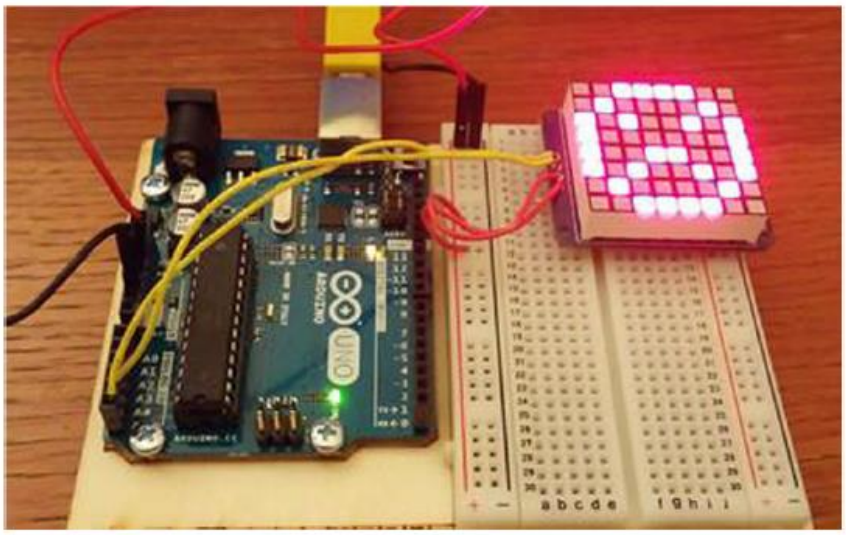

Imagen 3 placa Arduino usada para programación de patrones. Source: http://researchideas.ca/mc/article-1-title-recentissue/arduino-math-patterns-on-an-led-matrix/ (reproduced with permission)

Como Hughes et al. (2016) señala, hacer con Arduino tiene predominantemente un piso muy "alto" y se necesita más investigación para comprender y desarrollar herramientas que permitan un piso "bajo" y "alto". Esto es consistente con nuestra visión de hacer que las experiencias tengan un nivel de demanda cognitiva más alto que las experiencias desenchufadas o de hacer lúdico. Esto también ilustra que, a veces, la herramienta digital utilizada puede ser demasiado exigente desde el punto de

vista cognitivo para conceptos que pueden entenderse más simplemente a través de la selección de otras herramientas y experiencias.

4- Hacer con remezcla (remixing)

Las experiencias de remezcla se refieren a la apropiación de objetos o componentes de éstos para su uso en otros objetos u otros fines. La remezcla a veces se denomina "hacking". **La remezcla**, como se usa en este contexto, **es más que simplemente compartir un objeto o compartir y hacer una modificación menor en la que los objetos producidos son "en gran parte derivados, poco interesantes y de mala calidad"** (Dasgupta et al. 2016, p. 1439). Las experiencias de remezcla implican compartir (intencionalmente o mediante hacking) un objeto y modificarlo o adaptarlo de alguna manera y/o incrustarlo en otro objeto para usarlo con fines sustancialmente diferentes. **La remezcla requiere un nivel significativo de competencia para identificar un objeto utilizable y luego adaptarlo y modificarlo para los nuevos propósitos**. **Para remezclar, el estudiante debe poder ver objetos utilizables dentro de otros objetos. En nuestra opinión, esta es la tarea más exigente desde el punto de vista cognitivo y una que sugiere un avance sustancial a lo largo de una zona de desarrollo próximo**. En consecuencia, las experiencias de remezcla suceder después de las otras tres o varias combinaciones o iteraciones de las otras tres experiencias en el CTPF.

La remezcla es común en la comunidad informática, ya que los programadores a menudo utilizan el código de fuente abierto como una base y luego lo modifican para satisfacer sus necesidades particulares. Según Resnick et al. (2009), "los miembros de la comunidad están constantemente pidiendo prestado, adaptan y construyen sobre las imágenes, programas e ideas de otros. Más del 15% de los proyectos. ... son remezclas de otros proyectos "(p. 65). Las percepciones de la remezcla han evolucionado desde nociones negativas de que otros "roban" ideas, a una visión más abierta de una comunidad de aprendices que "se siente orgullosa, no molesta, cuando sus proyectos son adaptados y remezclados por otros " (Resnick et al. 2009, p. 65). Las comunidades de programación en línea ofrecen un beneficio claro para los estudiantes, ya que les permite construir sobre el trabajo de otros y también proporcionan un foro para dar crédito a otros cuando lo hacen (Watters, 2011b). La remezcla ofrece oportunidades importantes para participar en discusiones con los estudiantes sobre los problemas éticos asociados con la apropiación, la mezcla, la hibridez y el uso responsable de las tecnologías digitales (Colton 2016). Estas oportunidades para los estudiantes pueden servir para interceder en concesiones fuertemente utilitaristas sobre el pensamiento computacional y ampliar las discusiones críticas sobre las consecuencias de la computación generalizada. Estas oportunidades también pueden ayudar a construir la ciudadanía digital.

Dasgupta et al. (2016) probó la creencia de que la mezcla, particularmente la apropiación de medios programables en Scratch, promueve el aprendizaje (Lifelong Kindergarten Group en el MIT Media Lab 2016). Usando datos de más de un millón de cuentas de usuarios de Scratch, muestran cuidadosamente que "los usuarios que remezclan más a menudo tienen repertorios más grandes de comandos de programación [y que] la exposición a los conceptos de pensamiento computacional a través de la remezcla está asociada [positivamente] a una probabilidad mayor de usar esos conceptos" (p. 1438). Sugieren que el diseño de experiencias de remixes construidas deliberadamente junto con una mayor estructura en un entorno de aprendizaje informal como Scratch podría registrar efectos más grandes. También señalan que "la remezcla puede ser más efictivo para promover el compromiso con algunos conceptos, como los ciclos, que otros, como los operadores y los datos" (p.1446). De acuerdo con nuestra opinión de que la remezcla es más compleja en términos de desarrollo para los estudiantes, estos investigadores reconocen que "la remezcla requiere una participación prolongada y sigue siendo extremadamente difícil en los entornos de aprendizaje informal ... y que queda un enorme trabajo para alcanzar su potencial" (p.1446) .

Davis (2014) proporciona un ejemplo poderoso y unplugged de remixes en la educación matemática mediante el uso de analogía conceptual. Trabajando con los maestros en el tema de las operaciones matemáticas, Davis hizo que los maestros usaran sus experiencias previas y familiaridad con la suma, resta, multiplicación y división de “grillas” para generar una cuadrícula de exponenciación desde la cual surgió una rica conversación alrededor de las propiedades de la operación. Esta remezcla creativa, y no la mera reutilización de un artefacto popular, ilustra la forma en que la remezcla abre espacios para la elaboración recursiva.

Históricamente, hay evidencia que sugiere que los matemáticos utilizan habitualmente la remezcla para avanzar en la disciplina. El desarrollo del sistema de numeración hindú-árabe y el concepto de ángulo (Matos 1990, 1991) son dos ejemplos históricos. Más recientemente, el trabajo del matemático James Maynard sobre los números primos proporciona un buen ejemplo de remezcla en matemáticas (Freiberger 2016). Freiberger (2016) señala al describir el proceso emprendido por Maynard que:

> *"Este tipo de cosas es muy sintomático de las matemáticas modernas .... Muy a menudo se están vinculando muchas áreas diferentes de las matemáticas, y tal vez también se inspire en la física y la ingeniería, lo que normalmente no se habría pensado como asociado. Si dices 'números primos' nadie piensa en música y nadie piensa en cuboides con puntos en ellos. Pero resulta que todas estas cosas pueden estar relacionadas entre sí, y esta combinación de diferentes técnicas ha demostrado ser muy útil en matemáticas. (Párrafo 14) "*

Es probable que parte de la razón subyacente de que tales enfoques se hayan convertido en "sintomáticos de las matemáticas modernas" radique en la disponibilidad de entornos informáticos ricos para trabajar en matemáticas.

Ejemplo de CTPF

Usamos el concepto de una "variable" para proporcionar un ejemplo de las cuatro experiencias en el CTPF. Por definición, una variable se puede cambiar o adaptar de manera que varíe el resultado de una fórmula o proceso. Un ejemplo de una experiencia desenchufada que usa una variable podría ser cualquier juego no computarizado (por ejemplo, un juego de mesa) en el que un lanzamiento de dados determine el movimiento de los jugadores. Mientras que los límites inferior y superior de la variable están fijos a los valores de los dados, la cantidad, sin embargo, tiene el potencial de variar con cada tirada. De manera similar, los juegos que involucran tirar cartas o aterrizar en una casilla específica en un juego de mesa que luego tienen alguna instrucción desconocida o aleatoria o un valor que da forma a la jugada también son ejemplos de experiencias desenchufadas que involucran el uso de variables. Las cartas o el lugar en el tablero de juego son objetos tangibles que representan el concepto de variable. La medida en que el niño, o el jugador, reconoce la variable en la experiencia u objeto particular depende del grado en que la acción que representa la variable se hace explícita.

El mismo concepto de variable se ilustra fácilmente en una experiencia de hacer lúdico donde los estudiantes cambian un componente del código escrito previamente para modificar la salida. Por ejemplo, el código escrito previamente puede implicar mover un personaje una cierta cantidad de espacios hacia un objetivo final. El número de pasos se puede manipular en el programa cambiando el valor de la variable dentro de la estructura del programa. Al vincular la variable con la medición, los ángulos, el sentido numérico, etc., se conecta adicionalmente el aprendizaje con el contenido matemático. Las experiencias de hacer lúdico también pueden ser simultáneamente desenchufadas. Por ejemplo, una estructura preexistente que mueve agua, arena bolitas o vehículos, por ejemplo, pueden tener su camino alterado al cambiar interruptores. La dirección del interruptor es el componente variable de la ruta. Una experiencia de creación implicaría construir algún tipo de

programa u otro objeto tangible en el que se incluya un mecanismo de elección que pueda ser manipulado. Un interruptor electrónico que se activa de manera diferente según el evento anterior es un ejemplo en una experiencia de creación.

Finalmente, el desarrollo de código complejo a través de la remezcla podría implicar copiar o hackear programas o componentes de una estructura, explícitamente para hacer uso de variables (o estructuras que inducen la elección) que también podrían ser manipuladas aún más. En este ejemplo final, es probable que la variable esté incorporada en alguna otra estructura de programa que haga que la intervención final sea un proceso más eficiente que reescribir el código que contiene esa variable. Anteriormente proporcionamos ejemplos de cómo las fórmulas, que generalmente incluyen variables, se apropian en matemáticas para resolver problemas. En la remezcla, el alumno (o "hacker") debe comprender tanto la utilidad de la variable como el contexto actual y futuro en el que la variable realiza el cambio para aumentar la utilidad en su uso. Deben ser capaces de ver la variable y sus resultados, y esto desde nuestro punto de vista representa un nivel más alto de desafío cognitivo.

Estos ejemplos del concepto de variable desarrollado utilizando el CTPF representan un enfoque secuencial y también ilustran los casos en los que algunas experiencias son simultáneamente unplugged. El CTPF es útil especialmente para los principiantes en pensamiento computacional, ya que potencialmente proporciona una estructura pedagógica inicial. Puede darse el caso de que el punto de partida para algunos maestros esté en la etapa de “hacer lúdico”. Relacionar el concepto con experiencias desenchufadas, sin participar realmente en esas experiencias también puede ser útil. Del mismo modo, puede ser útil y necesario dar vueltas entre estas experiencias o incorporar simultáneamente versiones desenchufadas de las experiencias.

Proponemos que esto variará dependiendo del concepto, el nivel de conocimiento de los estudiantes y, probablemente, el nivel de comodidad pedagógica del maestro. El grado en que se produzcan tales casos y la forma en que tal disposición afectaría la enseñanza y el aprendizaje sería un área importante de investigación adicional.

Conclusiones

En este documento, proponemos e introducimos el CTPF que consta de cuatro experiencias: (1) hacer desenchufado, (2) hacer lúdico, (3) hacer creativo y (4) hacer con remezcla. El construccionismo (Papert 1987), y la teoría de Vygotsky sociocultural, específicamente la zona de desarrollo próximo de Vygotsky (1978), subrayan nuestro marco propuesto. CTPF **no pretende ser prescriptivo o necesariamente secuencial** y, aunque enfocamos nuestros ejemplos en educación matemática, proponemos que **el marco puede aplicarse más ampliamente para incluir otras disciplinas.**

El CTPF propuesto tiene como objetivo proporcionar un foco preliminar para estructurar la enseñanza y el aprendizaje para los estudiantes al considerar cómo enseñar el PC en lugar de qué enseñar o las prácticas asociadas con el PC que han sido descritas por otros (Brennan y Resnick 2012; Resnick et al. 2009 ; Resnick et al. 2005; Weintrop et al. 2016). A lo largo de todas las experiencias propuestas, los profesores deberían idealmente **construir lecciones y aprendizajes de tal manera que apoyen un piso bajo y un techo alto** (Papert 1980) pero también con lo que Resnick et al. (2009) llamado "**paredes 'anchas'**". .. ***apoyando muchos tipos diferentes de proyectos para que las personas con diferentes intereses y estilos de aprendizaje puedan participar*** "(p. 63).

Un desafío importante para el futuro será la capacitación de los maestros, tanto en servicio como en formación (Kafai 2015). El CTPF propuesto puede ser tremendamente útil para maestros y alumnos que pueden tener un entendimiento limitado del PC. Dado que este interés en el PC es un resurgimiento relativamente nuevo, no es probable que los maestros hayan encontrado formación/capacitación en el PC durante su formación (Curzon et al. 2014). En consecuencia, existe una "*brecha de conocimientos y habilidades, especialmente con respecto a la pedagogía basada en el*

sujeto" (Curzon et al. 2014, p. 89). Puede ser que los maestros en servicio y en formación se encuentren exactamente en el mismo punto de partida en términos de conocimiento e implementación y, por lo tanto, algunas sinergias en el desarrollo del maestro puedan ser posibles.

El CTPF propuesto también puede ser una estructura útil para apoyar el desarrollo docente. Curzon et al. (2014) **sugieren que los talleres para maestros también deberían comenzar con experiencias desenchufadas** para que se reduzca la brecha en el conocimiento del concepto de ciencias de la computación y tengan más confianza al usar las herramientas en su salón de clases. Un enfoque secuencial del CTPF, que comienza con experiencias desenchufadas y trabaja para remezclar experiencias, puede ser particularmente útil para los principiantes. Sin embargo, un enfoque secuencial no es esencial; más bien, asegurar la exposición a las cuatro experiencias en el CTPF puede ser más óptimo en términos de apoyar el desarrollo del PC.

Dada la naturaleza preliminar de nuestras afirmaciones sobre el CTPF, existen numerosas oportunidades para futuras investigaciones. Además de los estudios enfocados relacionados con las matemáticas y el PC, es necesario realizar una investigación interdisciplinaria relacionada con cada una de las cuatro experiencias para comprender las implicaciones para la enseñanza y el aprendizaje más allá de las aplicaciones tradicionales de la computación. Se necesita investigación para explorar el CTPF completo y su utilidad para el aprendizaje del PC. Dicha investigación podría explorar la eficacia del marco en relación con el desarrollo del conocimiento del PC a nivel de docentes y estudiantes y el desarrollo profesional del PC en docentes.

Referencias

Arduino (2016). Arduino. Retrieved August 14, 2016, from https://www.arduino.cc/.

Barr, V., & 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, M. (2013). Computing in the national curriculum. A guide for primary teachers. Bedford: Computing

at School.

Bers, M. U., & Horn, M. S. (2010). Tangible programming in early childhood. In R. Berson & M. J. Berson (Eds.), High tech tots: Childhood in a digital world (pp. 49–69). Charlotte: IAP.

Bowler, L. (2014). Creativity through "maker" experiences and design thinking in the education of librarians. Knowledge Quest, 42(5), 58–61.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Paper presented at the American Educational Research Association. Canada: British Columbia.

British Columbia Government. (2016). $6 million to help connect students with coding, new curriculum and computers. Retrieved August 11, 2016, from https://news.gov.bc.ca/releases/2016PREM0065-000994.

Statistics Canada. (2013). Canadian internet use survey, 2012. Retrieved June 29, 2015, from http://www.statcan.gc.ca/daily-quotidien/131126/dq131126d-eng.htm.

Colton, J. S. (2016). Revisiting digital sampling rhetorics with an ethics of care. Computers and Composition, 40, 19–31.

Corral, J. M. R., Balcells, A. C., Estévez, A. M., Moreno, G. J., & Ramos, M. J. F. (2014). A game-based approach to the teaching of object-oriented programming languages. Computers & Education, 73(83–92).

Curzon, P. (2013). cs4fn and computational thinking unplugged. WiPSE '13 Proceedings of the 8th Workshop in Primary and Secondary Computing Education, 47–50.

Curzon, P., McOwan, P., Plant, N., & Meagher, L. (2014). Introducing teachers to computational thinking

using unplugged storytelling. WiPSCE'14 Proceedings of the 9th Workshop in Primary and Secondary Computing Education, 89–92.

Dasgupta, S., Hale, W., Monroy-Hernandez, A., & Hill, B. M. (2016). Remixing as a pathway to computational thinking. Paper presented at the Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing.

Davis, B. (2014). Toward a more power-full school mathematics. For the Learning of Mathematics, 34(1), 12–17.

Dougherty, D. (2012). The maker movement. Innovations, 7(3), 11–14.

Farr, W., Yuill, N., & Raffle, H. (2010). Social benefits of a tangible user interface for children with autistic

spectrum conditions. Autism, 14(3), 237–252.

Freiberger, M. (2016). Primes without 7 [Electronic Version]. +plus magazine. Retrieved August 11, 2016,

from https://plus.maths.org/content/missing-7s.

Gadanidis, G. (2015). Coding as a Trojan horse for mathematics education reform. Journal of Computers in

Mathematics and Science Teaching, 34(2), 155–173.

Gadanidis, G., Hughes, J. M., Minniti, L., & White, B. J. G. (2016). Computational thinking, grade 1 students

and the binomial theorem [electronic version]. Digital Experiences in Mathematics Education.

doi:10.1007/s40751-016-0019-3.

Govender, I., & Grayson, D. J. (2008). Pre-service and in-service teachers' experiences of learning to program

in an object-oriented language. Computers & Education, 51(2), 874–885.

Government of England. (2013). National curriculum in England: Computing programmes of study. Retrieved

June 29, 2015, from https://www.gov.uk/government/publications/national-curriculum-in-englandcomputing-

programmes-of-study.

Horn, M. S., Crouser, R. J., & Bers, M. U. (2012). Tangible interaction and learning: The case for a hybrid

approach. Personal and Ubiquitous Computing, 16(4), 379–389.

Hoyles, C., & Noss, R. (2015). Revisiting programming to enhance mathematics learning, Math + Coding

Symposium. Western University: Western University. London.

Hughes, J., Gadanidis, G., & Yiu, C. (2016). Digital making in elementary mathematics education [electronic

version]. Digital Experiences in Mathematics Education. doi:10.1007/s40751-016-0020-x.

Kafai, Y. B. (2015). Connected code: A new agenda for K-12 programming in classrooms, clubs, and

communities. Paper presented at the Math + Coding Symposium: Western University, London.

Kafai, Y. B., & Burke, Q. (2013). Computer programming goes back to school. Phi Delta Kappan, 95(1), 61.

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and

programming workshop on sequencing ability in early childhood. Early Childhood Education Journal,

41(4), 245–255.

Kwon, D.-Y., Kim, H.-S., Shim, J.-K., & Lee, W.-G. (2012). Algorithmic bricks: A tangible robot programming

tool for elementary school students. IEEE Transactions on Education 55, 4(11), 474–479.

Lamagna, E. (2015). Algorithmic thinking unplugged. Journal of Computing Sciences in Colleges, 30(6), 45–

52.

Lambert, L., & Guiffre, H. (2009). Computer science outreach in an elementary school. Journal of Computing

Sciences in Colleges, 24(3), 118–124.

LeMay, S., Costantino, T., O'Connor, S., & ContePitcher, E. (2014). Screen time for children. IDC'14

Proceedings of the 2014 conference on Interaction design and children, 217–220.

Lifelong Kindergarten Group at the MIT Media Lab. (2016). Scratch. Retrieved August 11, 2016, from

https://scratch.mit.edu/.

Liu, C., Liu, K., Wang, P., Chen, G., & Su, M. (2012). Applying tangible story avatars to enhance children's

collaborative storytelling. British Journal of Educational Technology, 43(1), 39–51.

Lovell, E., & Buechley, L. (2011). LilyPond: An online community for sharing e-textile projects. New York:

Paper presented at the Proceedings of the 8th ACM conference on Creativity and Cognition.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through

programming: What is next for K-12? Computers in Human Behavior, 41, 51–61.

Matos, J. (1990). The historical development of the concept of angle. The Mathematics Educator, 1(1), 4–11.

Matos, J. (1991). The historical development of the concept of angle (2). The Mathematics Educator, 2(1), 18–

24.

Namukasa, I. K., Kotsopoulos, D., Floyd, L., Weber, J., Kafai, Y. B., Khan, S., et al. (2015). From

computational thinking to computational participation: Towards achieving excellence through coding in

elementary schools. In G. Gadanidis (Ed.), Math + coding symposium. London: Western University.

Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T., & Kuno, Y. (2009). A CS unplugged design

pattern. SIGCSE, 41(1), 231–235.

O'Sullivan, D., & Igoe, T. (2004). Physical computing: Sensing and controlling the physical world with

computers. Boston: Thomson.

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

Papert, S. (1987). Constructionism: A new opportunity for elementary science education. Retrieved August 1,

2016, 2016, from http://nsf.gov/awardsearch/showAward?AWD_ID=8751190.

Papert, S., & Harel, I. (1991). Constructionism: Ablex publishing corporation.

Parker, T. (2012). ALICE in the real world. Mathematics Teaching in the Middle School, 17(7), 410.

Pierce, M. (2013). Coding for middle schoolers: Next-generation programming languages for children are

taking up where Logo left off and teaching young students how to code to learn. T H E Journal

[Technological Horizons In Education], 40(5), 20+.

Province of Nova Scotia. (2015). Minister announces coding as a priority during education day. Retrieved

August 11, 2016, from http://novascotia.ca/news/release/?id=20151021002.

Przybylla, M., & Romeike, R. (2014). Physical computing and its scope - towards a constructionist computer

science curriculum with physical computing. Informatics in Education, 13(2), 225–240.

Resnick, M.,Myers, B., Nakakoji, K., Shneiderman, B., Pausch, R., Selker, T., et al. (2005). Design principles

for tools to support creative thinking. Washington DC: National Science Foundation workshop on

Creativity Support Tools.

Resnick,M.,Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch:

Programming for all. Communications of the ACM, 52(11), 60–67.

Scarlatos, L. L. (2006). Tangible math. Interactive Technology and Smart Education, 3(4), 293–309.

Shodiev, H. (2013). Computational thinking and simulation in teaching science and mathematics. Toronto:

Paper presented at the Association for Computer Studies Educators Conference.

SITRA. (2014). Future will be built by those who know how to code. Retrieved June 29, 2015, from

http://www.sitra.fi/en/artikkelit/well-being/future-will-be-built-those-who-know-how-code.

Smith, C. P., & Neumann, M. D. (2014). Scratch it out! Enhancing geometrical understanding. Teaching

Children Mathematics, 21(3), 185–188.

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Exploring the science framework and NGSS: Computational thinking in the science classroom. The Science Teacher, 38(3), 10–15.

Sphero. (2016). Retrieved August 11, 2016, from http://www.sphero.com/about.

Strawhacker, A., & Bers, M. U. (2015). "I want my robot to look for food": Comparing kindergartner's

programming comprehension using tangible, graphic, and hybrid user interfaces. International Journal of Technology and Design Education, 25(3), 293–319.

Sullivan, A., Kazakoff, E. R., & Bers, M. U. (2013). The wheels on the bot go round and round: Robotics curriculum in pre-kindergarten. Journal of Information Technology Education: Innovations in Practice, 12, 203–219.

Taub, T., Armoni, M., & Ben-Ari, M. (2012). CS unplugged and middle-wchool students' views, attitudes, and intentions regarding CS. ACM Transactions on Computing Education (TOCE), 12(2), 8.

The White House. (2016). Computer science for all. Retrieved August 11, 2016, from https://www.whitehouse.gov/blog/2016/01/30/computer-science-all

Thies, R., & Vahrenhold, J. (2013). On plugging Bunplugged^ into CS classes. SIGCSE '13 Proceeding of the 44th ACM technical symposium on computer science education, 365–270.

Vygotsky, L. S. (1978). Mind in society. Cambridge: Harvard University Press.

Watters, A. (2011a). Scratch: Teaching the difference between creating and remixing [Electronic Version]. Retrieved July 7, 2015, from http://ww2.kqed.org/mindshift/2011/08/11/scratch-teaching-kids-aboutprogramming-teaching-kids-about-remixing/.

Watters, A. (2011b). Scratch: Teaching the difference between creating and remixing 2015, from http://ww2.kqed.org/mindshift/2011/08/11/scratch-teaching-kids-about-programming-teaching-kids-about-remixing/.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1),127–147.

Wilkerson-Jerde, M. (2014). Construction, categorization, and consensus: Student generated computational artifacts as a context for disciplinary reflection. Educational Technology Research and Development, 62(1), 99–121.

Wing, J. M. (2006). Computational thinking and thinking about computing. Communications of the ACM, 49, 33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A, 366, 3717–3725.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing computational thinking in education courses. SIGCSE, 11, 465–470.

Yiu, C. (2016). Using an Arduino - coding a bicolour LED grid to create math patterns [electronic version]

(p. 1). Math + Coding 'Zine: Exploring Math Through Code Retrieved August 16, 2016, from

http://researchideas.ca/mc/article-1-title-recent-issue/arduino-math-patterns-on-an-led-matrix/.