a chave é a aprendizagem baseada em projetos. Esta metodologia de trabalho permite partir de um problema abrangente que pode ser trabalhado a partir de múltiplas disciplinas e buscar estratégias cooperativas para o alcance da aprendizagem comunitária. Professores de diferentes disciplinas de uma escola secundária trabalham na implementação de projetos interdisciplinares com foco no Egito Antigo. Por meio do trabalho cooperativo, os alunos resolveram desafios matemáticos, interpretaram textos, elaboraram infográficos e dançaram em um cenário ambientado com música e recursos digitais que estimularam a criatividade.

Palavras chave: Aprendizagem ativa - avaliação formativa - história - interdisciplinaridade - literatura - matemática - método de aprendizagem - trabalho colaborativo.

(*) Liliana Alicia Gutte. Profesora en Lengua y Literatura – Ciclo de Profesorado (UNR, Facultad de Humanidades y Artes). Postítulo de Formación universitaria en Lengua y Literatura (UNR, Facultad de Humanidades y Artes). Docente de Historia de la Lengua y Literatura Latinoamericana I (Instituto Sedes Sapientiae). Es docente de nivel secundario en el Instituto Malvina Seguí de Clavarino y el Instituto Sagrado Corazón. Autora del libro La reescritura del episodio de Juan Díaz de Solís: historia y relato, EAE, 2018. • **Gloria Fabiana Ríos** Profesora de Historia - Instituto de Profesorado "Sedes Sapientiae" Gualeguaychú. E.R. Docente de Historia y Formación Etica y Ciudadana en Instituto Malvina Seguí de Clavarino, nivel secundario. Docente de Historia y Formación Ética y Ciudadana en Escuela Secundaria Nº 12 Luis Clavarino. Docente en la asignatura Historia Económica Contemporánea en el profesorado de Economía del Instituto Sedes Sapientiae • **Carla Patricia Sánchez**. Profesora de Matemática (ISFD "María Inés Elizalde"). Especialización Docente de Nivel Superior en Educación y Tic, Especialización Docente de Nivel Superior en Enseñanza de la Matemática en la escuela secundaria y Especialización en Investigación Educativa. Es docente de nivel secundario (Instituto "Malvina Seguí de Clavarino"), docente I.S.P.E.D., docente en la Escuela Normal Superior "Olegario Víctor Andrade" y en el Instituto Superior de Arte.

Computación en Ingeniería: rol del diseño en el pensamiento computacional

Fecha de recepción: julio 2021
Fecha de aceptación: septiembre 2021
Versión final: noviembre 2021

Elizabeth Jiménez Rey (*)

Resumen: El desarrollo del pensamiento computacional (*computational thinking*) constituye la competencia clave de aprendizaje en el proceso de solución de problemas con la computadora (mediante el ingenio de algoritmos y su codificación como programas). Para diseñar un algoritmo efectivo, el estudiante puede aplicar la técnica de diseño descendente (*top-down*) o ascendente (*bottom-up*). Así, se posibilita la libre reflexión *(how to thinkit)* en el proceso de construcción de un algoritmo, mediante la descomposición en subproblemas (análisis) o la composición de subproblemas (síntesis). Ambas estrategias, promueven la metacognición del modo propio de solucionar un problema, y estimulan un aprendizaje consciente en el estudiante.

Palabras clave: Aprendizaje significativo - enseñanza - diseño.

[Resúmenes en inglés y portugués en la página 201]

Diseño de soluciones y pensamiento computacional

La enseñanza de Computación en Ingeniería se focaliza en la solución de problemas con la computadora. En la Facultad de Ingeniería de la Universidad de Buenos Aires, la asignatura es de formación básica por Resolución Ministerial del año 2001, es obligatoria para los estudiantes de las carreras de ingenierías en general y es cuatrimestral, es decir, se desarrolla en dieciséis clases, una por semana, de cuatro horas de duración.

Los estudiantes deben desarrollar la capacidad de construir la solución de problemas con la computadora, mediante el ingenio de algoritmos y su codificación como programas. El mayor desafío cognitivo al que se enfrentan los alumnos es el descubrimiento de un algoritmo efectivo, eficaz y eficiente, que solucione el problema planteado.

En el enfoque procedimental se enfatiza no solamente el *saber hacer* sino también el *saber por qué se hace*. Se abordan los contenidos desde tres núcleos centrales -algoritmo, programa y computadora- los cuales se interrelacionan y retroaccionan entre sí. El eje estructurante del curso se organiza en espiral, y en cada clase, el descubrimiento de algoritmos y el desarrollo de programas se complejiza, y el conocimiento de la computadora se profundiza.

Los estudiantes utilizan el Modelo de Solución de Problemas del matemático húngaro George Polya, aplicado al ámbito de la construcción de programas. El Modelo consta de cuatro fases: Análisis (comprender el problema), Diseño (diseñar una estrategia), Codificación (implementar la estrategia) y Evaluación (ejecutar el programa). El Análisis del Problema y el Diseño de la

Solución, son las *fases algorítmicas* y constituyen el desafío creativo de aprendizaje. La Codificación del Algoritmo y la Evaluación del Programa son las *fases de programación*. La mayor dificultad para los estudiantes reside en ingeniar un algoritmo que solucione el problema. En Computación, el *algoritmo* es el procedimiento que permite resolver un problema y el *programa* es la expresión del algoritmo en un lenguaje de programación para que la computadora pueda ejecutarlo.
En general, se focaliza el aprendizaje de la construcción de programas en las fases de programación, es decir, en que los estudiantes puedan implementar una secuencia lógica de instrucciones en un lenguaje de programación para luego ejecutar el programa con valores de prueba y comprobar su correcto funcionamiento. Y en consecuencia, los estudiantes, alcanzan este objetivo por prueba y error, de forma inconsciente y automática, secuenciando instrucciones para solucionar problemas de dificultad progresiva, sin analizar la naturaleza del problema y sin detectar cuáles son las herramientas de programación apropiadas para implementar la solución. Así, se reduce el proceso evolutivo de construcción de programas, a las fases de programación, Codificación y Evaluación, disociándolas de las fases algorítmicas, Análisis y Diseño.
Se favorece el desarrollo del pensamiento computacional (*computational thinking*) mediante el análisis del problema y el diseño de algoritmos, como una competencia clave de aprendizaje en la formación de los estudiantes de Ingeniería. El descubrimiento de un algoritmo es un proceso creativo. No hay un método *a priori* para idear un algoritmo. Se concibe el método como camino, como ensayo generativo y estrategia para el pensamiento. El método como ensayo se organiza en *tablas,* a modo de expresión escrita del pensamiento y de la reflexión. Las *tablas* proporcionan una estructura para que el estudiante desarrolle su capacidad de crear y de aprender en la búsqueda de la transformación de los datos en resultados. Constituyen un andamiaje al pensamiento computacional y posibilitan la implementación de las fases algorítmicas.

Diseño descendente y Diseño ascendente
Para diseñar un algoritmo efectivo, se promueve la aplicación de la técnica de diseño descendente (*top-down*), es decir, la descomposición *en* subproblemas del problema, hasta llegar al último nivel de descomposición (acciones esenciales). Se diseñaron *tablas* para ayudar a los estudiantes a descubrir algoritmos que solucionan los problemas propuestos:

• Tabla 1. Definición del Objetivo y de los Recursos. Los estudiantes se focalizan en las especificaciones del enunciado del problema para aislar sus elementos y clasificarlos según sus características. Realizan una *lectura comprensiva* del enunciado (¿qué problema se debe resolver?) y una *lectura de rastreo* del enunciado (¿cuáles son los recursos necesarios para resolver el problema?). Se orienta y visibiliza el pensamiento mediante descripciones y preguntas guía que proporcionan distintas perspectivas de análisis para la elaboración de la respuesta.

• Tabla 2. Análisis del Problema. Los estudiantes se focalizan en las especificaciones del enunciado del problema como conocimiento previo para seleccionar los casos representativos del problema, establecer el estado inicial (datos) y el estado final (resultados) del problema y determinar los estados intermedios de transformación (procedimiento aritmético) desde el estado inicial hasta el estado final.
• Tabla 3. Diseño de la Solución. Los estudiantes amplían el conocimiento del problema y avanzan a la primera columna de esta tabla considerando la última columna de la tabla anterior para determinar el estado genérico de adaptación computacional, realizar la descomposición del problema en subproblemas, detectar la naturaleza de los subproblemas y seleccionar las primitivas de programación apropiadas en lenguaje *Python* para implementar cada uno de los subproblemas.

Los estudiantes utilizan las *tablas* y trabajan en pequeños grupos de tres integrantes en talleres virtuales propios creados en la plataforma *Google Drive* Institucional denominados *Un Lugar para Algoritmiar*. Durante el proceso de diseño de la solución, los estudiantes hacen visible el pensamiento computacional, acompañados por el profesor quien realiza el control de la calidad de diseño de la solución algorítmica.
A continuación, utilizan la Tabla 4, Modelo de Programa Tipo, en el entorno de desarrollo y aprendizaje IDLE Python denominado *Un Lugar para Programar*. La tabla organiza el texto del programa (algoritmo codificado, momento estático) en secciones que implican las cuatro fases del proceso de construcción del programa. En el último nivel de descomposición de los subproblemas, las acciones esenciales algorítmicas (enunciados no ejecutables) se convierten en instrucciones de programa (primitivas de programación) en lenguaje *Python*. Las descripciones de los subproblemas a implementar con las instrucciones forman parte del texto del programa y constituyen los enunciados de documentación interna del programa. Se ejecuta en la computadora el algoritmo codificado con valores de prueba para comprobar el correcto funcionamiento del programa (programa ejecutado, momento dinámico).
Así, se distinguen y se conjugan las *fases algorítmicas* y las *fases de programación* en el proceso evolutivo de construcción de programas.
Si los estudiantes deciden iniciar el proceso de construcción del programa para solucionar el problema propuesto, desde la fase de Codificación y, trabajar en primer lugar en el espacio Un Lugar para Programar, es decir, codificar directamente la solución en lenguaje *Python*, secuenciando las primitivas de programación y evaluando el funcionamiento del programa con valores de prueba de los datos, el requerimiento del logro de un programa inteligible, no solamente eficaz y eficiente, posibilita la reflexión sobre el proceso de diseño de la solución. El programa editado a entregar debe contener los enunciados de documentación interna. Por lo tanto, el estudiante debe realizar un proceso inverso para lograr la composición *de* subproblemas que integran la solución del problema propuesto. Será necesario ana-

lizar qué subproblema resuelve una instrucción o un conjunto de instrucciones, reflexionar sobre el modo propio de solucionar el problema, sintetizar el pensamiento computacional, es decir, aplicar una técnica de diseño ascendente (*bottom-up*). Y completar las *tablas*, siguiendo un proceso inverso que le permitirá visibilizar las modificaciones a introducir en el diseño del algoritmo codificado para mejorar la calidad de la implementación de la solución.

Enseñanza activa y Aprendizaje consciente

Las *tablas*, como andamiaje para el desarrollo del pensamiento computacional y el diseño del algoritmo que soluciona el problema, ayudan a los estudiantes a pensar *(how to think it)* para enfrentar la incertidumbre y la perplejidad que se presentan cuando deben abordar la solución de un problema con la computadora.

Las *tablas* son dinámicas porque las orientaciones del profesor en cada columna se adaptan a las características del problema a resolver. Cada columna de cada tabla define una acción a desarrollar. Los estudiantes completan las *tablas* acompañados por el profesor quien interviene en el proceso creativo para modelar el pensamiento.

Ambas estrategias de diseño, descendente o ascendente, promueven la metacognición del modo propio de solucionar un problema, y estimulan un aprendizaje consciente en el estudiante. El desarrollo del pensamiento computacional se implementa, más allá de construir programas, con el propósito de enseñar a pensar en Computación. Se intenta establecer puentes con el desarrollo del pensamiento ingenieril, necesario para aprender a pensar en Ingeniería, vinculando en el proceso formativo de los estudiantes, Enseñanza de Computación y Educación en Ingeniería.

Referencias bibliográficas

Jiménez Rey, E. (2011). *Enseñanza y Aprendizaje de Computación en Carreras de Ingeniería: una visión sistémica de los procesos esenciales y sus vinculaciones con las NTICs*. I Congreso Argentino de Tecnología de Información y Comunicaciones. Las Nuevas Tecnologías en la Sociedad de la Información y del Conocimiento. Buenos Aires: COPITEC.

Jiménez Rey, E., Aveleyra, E. y Barranquero, F. (2019). *Competencias en algoritmia y programación como formación básica en ingenierías: El rol del pensamiento visible y de la mediación tecnológica.* Actas de Resúmenes IV Congreso Internacional de Enseñanza de las Ciencias Básicas, pp. 14-15. Paysandú, Uruguay.

Jiménez Rey, E. (2019). *Computación en Ingeniería: La experiencia de pensar para solucionar problemas con algoritmos y programas en aula real y aula virtual.* Anales de SAEI 2019. Simposio Argentino de Educación en Informática (JAIIO), ISSN 2683-8958, pp. 1-16.

Morin, E. (2008). *La cabeza bien puesta. Repensar la reforma. Reformar el pensamiento.* Buenos Aires: Nueva Visión.

Nickerson, R. S., Perkins, D. N. y Smith, E. E. (1987). *Enseñar a pensar. Aspectos de la aptitud intelectual.* Barcelona: Paidós.

Ritchhart, R., Church, M. y Morrison, K. (2014). *Hacer visible el pensamiento.* Buenos Aires: Paidós.

Simari, G. (2013). *Los fundamentos computacionales como parte de las ciencias básicas en las terminales de la disciplina Informática*. VIII Congreso de Tecnología en Educación y Educación en Tecnología. Santiago del Estero.

Swartz, R.; Costa, A.; Beyer, B.; Reagan, R. y Kallick, B. (2008). *El aprendizaje basado en el pensamiento. Cómo desarrollar en los alumnos las competencias del siglo XXI*. Nueva York: Ediciones SM.

Wing, J. (2006). "Computational Thinking". En: *Communications of the ACM*. Marzo. Vol. 49, n°. 3, 33-35.

Wing, J. (2010). *Computational Thinking: What and Why?* Recuperado de: https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Zapata Ros, M. (2015). *Pensamiento Computacional: Una nueva alfabetización digital*. RED Revista de Educación a Distancia, 46(4). Recuperado de: https://www.um.es/ead/red/46/

Abstract: The development of computational thinking constitutes the key learning competence in the process of solving problems with the computer (through the ingenuity of algorithms and their codification as programs). To design an effective algorithm, the student can apply the top-down or bottom-up design technique. Thus, free reflection (how to thinkit) is made possible in the process of building an algorithm, through the decomposition into subproblems (analysis) or the composition of subproblems (synthesis). Both strategies promote metacognition of the proper way of solving a problem, and stimulate conscious learning in the student.

Keywords: Meaningful learning - teaching - design.

Resumo: O desenvolvimento do pensamento computacional (*computational thinking*) constitui a competência chave de aprendizagem no processo de resolução de problemas com o computador (através da engenhosidade dos algoritmos e sua codificação como programas). Para projetar um algoritmo eficaz, o aluno pode aplicar a técnica de design decima para baixo (*top-down*) ou de baixo para cima (*bottom-up*). Assim, a reflexão livre *(how to thinkit)* é possível no processo de construção de um algoritmo, seja pela decomposição em subproblemas (análise) ou pela composição de subproblemas (síntese). Ambas as estratégias promovem a metacognição da maneira adequada de resolver um problema e estimulam a aprendizagem consciente do aluno.

Palavras chave: Aprendizagem significativa - ensino - design.

(*) Elizabeth Jiménez Rey. Docente Investigador en el Departamento de Computación de la Facultad de Ingeniería de la Universidad de Buenos Aires. Ingeniero Civil. Especialista en Ingeniería de Sistemas. Diplomada en Inteligencia y Pedagogía Compleja. Maestranda en Tecnología Informática Aplicada en Educación en la Facultad de Informática de la Universidad Nacional de La Plata. Área de Docencia: Computación. Área de Investigación: Educación en Tecnología y Tecnología en Educación.