TEC | Tecnológico de Costa Rica

Informe Final

Documento 1

Proyecto de Investigación

Metodología para el reconocimiento automático de patrones del pensamiento computacional en estudiantes de la educación general básica para mejorar los procesos de gestión

Código 1370013

Equipo de investigación

Dra. Lilliana Sancho Chavarría
coordinadora

Dr. Cesar Garita Rodríguez
Dr. Antonio González Torres
Dr. Jorge Monge Fallas
MSc. José Navas Sú

Escuela de Ingeniería en Computación
Centro de Investigaciones en Computación
8 de febrero, 2024

Tabla de contenido

1. Autores y direcciones

Dra.-Ing. Liliana Sancho Chavarría	lsancho@itcr.ac.cr
Dr. César Garita Rodríguez	cesar@itcr.ac.cr
Dr. Antonio González Torres	antonio.gonzalez@itcr.ac.cr
Dr. Jorge Monge Fallas	jomonge@itcr.ac.cr
MSc. José Navas Sú	jnavas@itcr.ac.cr

2. Resumen

El pensamiento computacional (PC) refiere al pensamiento crítico, la capacidad de razonamiento y de solución de problemas. Algunas referencias en la literatura centran el desarrollo del PC en la estimulación de esas habilidades a través de la programación de computadoras, otras la conciben de forma más amplia a través de experiencias que estimulen esas capacidades. Aplicar actividades que fomenten el desarrollo del PC supone un conjunto de ventajas para el aprendizaje y para que las personas enfrenten los retos de la sociedad.

El objetivo general de esta investigación consistió en definir una metodología para comprender la correlación entre los objetivos de aprendizaje y las habilidades de PC, tomando como insumo el código fuente de los programas elaborados por estudiantes del Programa de Informática Educativa del MEP-FOD.

El método de investigación utilizado se enmarca en análisis de contenido, el cual considera el análisis de patrones en soluciones a ejercicios de programación desarrollados por estudiantes de primaria.

Producto de esta investigación se destacan los siguientes resultados: diseño de un framework para el análisis de código fuente, algoritmos que utilizan técnicas de "clustering" para identificar patrones de solución en ejercicios programados, algoritmos para la elaboración y evaluación de ejercicios, visualizaciones para el

análisis de datos, cuatro artículos científicos publicados (indexación Scopus), dos artículos científicos por publicar.

Concluimos que es posible identificar patrones de PC en términos de la capacidad para la representación de datos, flujo de control, abstracción, paralelismo y complejidad ciclomática. Además, logramos identificar requisitos de los conjuntos de datos (datasets) para futuros análisis e investigación y confirmamos que el análisis de código fuente es un enfoque novedoso en investigación de PC.

3. Palabras clave

Pensamiento computacional, resolución de problemas, aprendizaje, análisis automático de código fuente, evaluación educativa, automatización de procesos

4. Introducción

La formación de PC es un proceso de transformación del razonamiento de los estudiantes que conlleva la adquisición de habilidades para solucionar problemas concretos y abstractos, en condiciones de ambigüedad, tomando como base una serie de pasos detallados, hasta alcanzar una solución. Costa Rica tiene la particularidad de que desde el año 1988 cuenta con el Programa Nacional de Informática Educativa del Ministerio de Educación Pública y la Fundación Omar Dengo (PRONIE MEP-FOD), mediante el cual introdujo las tecnologías digitales en las escuelas costarricenses, como una estrategia para potenciar el desarrollo de capacidades cognitivas y sociales en los estudiantes. El PRONIE-MEP-FOD busca estimular competencias tales como la resolución de problemas, mediante un enfoque de aprendizaje por proyectos usando la programación como estrategia didáctica, y fundamentado en el marco epistemológico del construccionismo social de Seymor Papert, basado en la teoría constructivista de Piaget (FOD, 2019). En la propuesta educativa para los Laboratorios de Informática Educativa,

denominada "*LIE++: pensar, crear, programar*", en el año 2018 se incluye el enfoque de Pensamiento Computacional (PC), considerada una tendencia mundial para el desarrollo de habilidades. Es así como, desde un enfoque centrado en el estudiante, el trabajo propuesto a desarrollar con LIE++ se orientó al desarrollo de competencias tales como: resolución de problemas con programación, manejo de operaciones y componentes de sistemas computacionales, representación y modelaje de datos, construcción de artefactos físicos y robots y comunicación y colaboración para explicar y construir de forma colaborativa. De esta manera, para el PRONIE MEP-FOD, el Pensamiento Computacional se define como la habilidad de resolver problemas apoyándose en los fundamentos, conceptos y prácticas de la computación para crear soluciones a los problemas, que pueden concretarse en representaciones programadas automatizadas, que podrían variar desde una representación digital, hasta un prototipo físico que integra sensores y actuadores. (FOD, 2018).

La gestión del proceso de enseñanza y aprendizaje de PC en la educación en general, y particularmente en primaria secundaria, requiere información adecuada para planificar y evaluar los planes de estudio, los objetivos de aprendizaje y los métodos de evaluación. El conocimiento generado por la FOD por más de treinta años constituye una ventaja comparativa por la experiencia acumulada, y por contar con una base de datos de código fuente de ejercicios y proyectos desarrollados por estudiantes, los cuales podrían ser analizados automáticamente para determinar patrones de pensamiento computacional en las soluciones, con el propósito de mejorar los procesos de gestión.

La revisión de literatura permitió determinar la falta de investigaciones, métodos y herramientas que estudien código fuente, producto de ejercicios resueltos en el contexto educativo, para analizar pensamiento computacional, siendo la presente una investigación innovadora. Los nuevos métodos propuestos serían validados

mediante un caso de estudio y están orientados a beneficiar a los estudiantes costarricenses y a la sociedad en general.

El método de investigación utilizado consiste en el análisis de contenido de la base de datos de ejercicios programados de la FOD. Se realizó el análisis exhaustivo de los ejercicios desarrollados por los estudiantes, de diferentes períodos lectivos y regiones del país. En esta investigación se aplicaron métodos de análisis sintáctico, análisis semántico, aprendizaje automático y teoría matemática de lenguajes de programación para efectuar la evaluación automática de la base de ejercicios de programación con el fin de apoyar acciones para mejorar la gestión de los programas de enseñanza de PC.

5. Marco Teórico

El término *pensamiento computacional* (en adelante PC) fue acuñado por Jeannett Wing en al año 2006 (Wing, 2006), quien lo definió como un proceso que involucra resolver problemas, diseñar sistemas y entender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática. Unos años más tarde, Wing ajusta su definición estableciendo que el pensamiento computacional son los procesos de pensamiento que intervienen en la formulación de los problemas y sus soluciones, de modo que éstas se representen de forma que puedan ser llevadas a cabo eficazmente por un agente procesador de información (Wing, 2008). En el año 2011 la Computer Science Teachers Association (CSTA) y la International Society for Technology in Education (ISTE) definen pensamiento computacional como un proceso de solución de problemas que incluye (aunque no está limitado a) las siguientes características: formular problemas de un modo que se haga posible utilizar un ordenador y otras máquinas en su resolución; organizar lógicamente y analizar datos; representar datos a través de abstracciones tales como modelos y simulaciones; automatizar soluciones a

través del pensamiento algorítmico (una serie de pasos discretos y ordenados); identificar, analizar e implementar posibles soluciones con el objetivo de lograr la combinación más efectiva y eficiente de pasos y recursos; generalizar y transferir este proceso de solución de problemas a una amplia variedad de problemas. Específicamente, la definición operacional propuesta por ISTE y CSTA tiene seis características: abstracción, recopilación y análisis de datos, representación de datos, solución automática, combinación efectiva de pasos y recursos, y solución de transferencia (Seehorn, 2011). Por otro lado, Román-González considera que el pensamiento computacional es la capacidad de formular y solucionar problemas apoyándose en los conceptos fundamentales de la computación, y usando la lógica-sintaxis de los lenguajes informáticos de programación: secuencias básicas, bucles, iteraciones, condicionales, funciones y variables (Román-González, 2015).

El proceso de desarrollo de PC requiere el uso de la abstracción, planificación y la descomposición de los problemas en las partes que los constituyen, con frecuencia, en la presencia de incertidumbre. PC se define como el proceso mediante el cual se especifican modelos computacionales para la formulación y solución de problemas mediante la especificación de pasos y algoritmos (Aho, 2012).

El desarrollo de PC en los estudiantes permite el aprendizaje de habilidades para la especificación precisa y el análisis de problemas, incentiva tomar en cuenta diferentes alternativas para solucionar un problema (normales y excepcionales) y facilita determinar las acciones necesarias para resolver un ejercicio de forma adecuada (Futschek, 2006). Además, fomenta el pensamiento crítico, la creatividad y la cooperación entre los alumnos (Doleck, Bazelais, Lemay, Saxena, & Basnet, 2017).

Los beneficios que produce el desarrollo del Pensamiento Computacional ha incentivado a diferentes organizaciones de países, como Costa Rica, Estados

Unidos, Holanda y el Reino Unido, para que se incluya como un elemento central en los currículos educativos de primaria y secundaria (Aman, Jon, Joke, & Fisser, 2017). En el caso concreto de Costa Rica, en el año 1988 se creó el Programa Nacional de Informática Educativa (PRONIE MEP-FOD) que tiene una cobertura superior al 92% de la población estudiantil de centros educativos públicos. El objetivo de dicho programa es fomentar el desarrollo de las habilidades para resolver problemas (Zúñiga-Céspedes, 2018).

En línea con lo anterior, el estudio de la bibliográfica relacionada permitió establecer la existencia de diferentes enfoques para apoyar a los docentes en la calificación automática de tareas y pruebas de programación durante el proceso de enseñanza de PC (Ala-Mutka, 2005; Chen, Chen, Hsueh, Lee, & Li, 2017), los cuales, por lo general, se han aplicado de forma limitada a uno o varios cursos que son impartidos usando un lenguaje de programación particular (Poženel, Fürst, & Mahnič, 2015). Sin embargo, no fue posible identificar trabajos de investigación que se concentren en la evaluación automática del cumplimiento de los objetivos de aprendizaje por parte de los estudiantes y docentes, a nivel integral de centros educativos o escala nacional (Cheang, Kurnia, Lim, & Oon, 2003).

Los métodos utilizados en la literatura para realizar la calificación automática o semiautomática de ejercicios de programación contemplan el análisis estático y dinámico de código (Caiza & del Alamo, 2013), pruebas de unidad (Pawelczak, Baumann, & Schmudde, 2015), técnicas de “machine Learning” (Morales-Castellanos, 2017) o una combinación de éstas. Entre los aspectos que han sido considerados se encuentran los que examinan la funcionalidad, eficiencia, estilo de codificación, errores de programación y diseño de los programas (Ala-Mutka, 2005).

Sin embargo, los resultados de las investigaciones sobre la calificación automática de ejercicios no se encuentran disponibles de forma pública para reutilizar el software resultante, y las pocas herramientas disponibles no permiten el análisis

masivo de ejercicios de programación para verificar el cumplimiento de los objetivos de aprendizaje con el fin de definir políticas y los contenidos de los currículos formativos tomando como base los resultados. Por lo que se requiere proponer nuevos métodos y efectuar mejoras a los existentes, de acuerdo con las publicaciones, con el fin de proponer técnicas escalables que brinden mayor confiabilidad y efectividad para ser utilizadas en el análisis automático de grandes bases de ejercicios de programación para ofrecer información que permita apoyar acciones que contribuyan a mejorar la gestión de los programas de enseñanza de PC tanto a nivel nacional como internacional.

Entre los métodos que se estarán considerando durante el desarrollo de esta investigación se encuentran los orientados a la detección de clones de código Tipo-1, Tipo-2, Tipo-3 y Tipo-4 (Baker, 1992; Kamiya, Kusumoto, & Inoue, 2002), el cálculo de la complejidad ciclomática de los programas (McCabe, 1976) y el índice Halstead (Halstead, 2003). Los clones de código son copias de fragmentos de programas con una sintaxis idéntica o similar, o funcionalidad equivalente, pero que cuentan con una sintaxis diferente (Roy, Cordy, & Koschke, 2009; Wagner, Abdulkhaleq, Bogicevic, Ostberg, & Ramadani, 2016).

Detección de clones de código

Los clones de código se clasifican en cuatro tipos:

Tipo-1: Son fragmentos de código idénticos con varios espacios en blanco, los comentarios y la forma como se encuentran ordenados.

Tipo-2: Consisten en fragmentos sintácticos idénticos que tienen variaciones en el nombre de los identificadores, valores literales, tipos de variables, espacios en blanco y comentarios.

Tipo-3: También son fragmentos de código copiados, pero tienen más variantes como líneas agregadas o borradas y variaciones en los identificadores, literales, tipos, espacios en blanco y comentarios.

Tipo-4: Estos fragmentos de código son los más difíciles de identificar, porque realizan las mismas operaciones, pero son implementados usando diferentes variantes sintácticas. Este tipo de clones también reciben el nombre de clones funcionales.

La complejidad de la identificación de clones está asociada al análisis sintáctico y semántico de los fragmentos de código, pero también al proceso de búsqueda y la amplitud del espacio que se debe explorar. Lo anterior se debe a que un clon se puede componer de unas pocas líneas de código que deben ser comparadas de forma sucesiva con miles o incluso millones de líneas en archivos o clases diferentes durante el proceso de búsqueda.

Entre los enfoques más utilizados para efectuar la identificación y localización de clones se encuentran la minería de texto, enfoques basados en análisis de texto y tokens, Árboles de Sintaxis Abstracta (AST), grafos de dependencia de programas, análisis semánticos y técnicas de machine learning que usan aprendizaje supervisado. Entre los retos de este tipo de métodos se encuentra efectuar el procesamiento en el menor tiempo posible mediante la adaptación o diseño de algoritmos para plataformas de altas prestaciones (computación paralela). Sin embargo, un reto común que se debe afrontar para usar cualquiera de estas técnicas es contar con un conjunto de datos robusto para validar sus resultados, o en el caso de los métodos de aprendizaje supervisado, para entrenar los algoritmos y probar su eficacia.

Como consecuencia, la identificación de clones tipo 4, los más complejos, en el contexto de este trabajo requiere diseñar e implementar:

1. Un método basado en programación genética machine learning y técnicas de computación de altas prestaciones para crear de forma automática conjuntos de clones de código Tipo-4 para entrenar y/o validar que los algoritmos de identificación y localización de fragmentos de código equivalente lo hacen de forma correcta.

2. Un algoritmo para la identificación de clones Tipo-4 y su correspondiente validación al verificar que son funcionalmente equivalentes y no corresponden a clones Tipo-1, Tipo-2 y Tipo-3.

Complejidad Cinclomática

El resultado del cálculo de la complejidad ciclomática (McCabe, 1976) es la suma del número de puntos de decisión en el código, de forma que entre más alto es el resultado más complejo es el código. Por lo que el cómputo de esta métrica se centra el análisis estático del código y su representación, en la cual las sentencias son representadas como nodos y los caminos de control entre una sentencia y otra es una arista.

La fórmula para el cálculo de la complejidad ciclomática, comúnmente referenciada como CC, es la siguiente:

CC = E-N+2, en donde E es el número de aristas y N es el número de nodos.

Índice deHalstead

Por su parte, el índice de Halstead es una medida que considera todos los operadores y operandos distintitos en una pieza de código para identificar sus relaciones. En tanto, el Índice de Mantenibilidad de un programa (Coleman, Ash, Lowther, & Oman, 1994) es un modelo de regresión que se aplica sobre el volumen de Halstead, la complejidad ciclomática y el número de líneas de código.

El uso combinado de estas métricas permitirá detectar por un lado la equivalencia entre la solución de los ejercicios en función de los objetivos de aprendizaje y las respuestas ofrecidas, y por otro lado determinar si las resoluciones cumplen con lo esperado en términos de complejidad, tanto por el número de ciclos, puntos de decisión, operadores y líneas de código.

6. Metodología

a) Población:

La FOD suministró información de referencia de 3,953 centros educativos, los cuales tenían activo el Programa de Informática Educativa (PRONIE), y una cobertura de 740,456 estudiantes de primaria (379,919 niños y 360,537 niñas) en zonas urbanas y rurales, de las siete provincias. Los datos incluyen, entre otros, código y nombre del centro educativo, fecha de inicio en el PRONIE, localización, provincia, cantón, distrito, poblado, dirección exacta, tipo de zona (rural/urbana), circuito escolar, cantidad de secciones, cantidad de hombres, cantidad de mujeres beneficiados.

La FOD facilitó el acceso al repositorio GECO en el cual se almacenan las soluciones de código fuente generado por los estudiantes de los Laboratorios de Informática Educativa (LIE). GECO se empezó a alimentar con datos en el año 2018, por lo cual, al inicio de este proyecto de investigación, la FOD había almacenado en el repositorio 2 años de datos, se esperaba para los 2020, 2021 y 2022 ir contando con mayor cantidad de datos; no obstante, las restricciones sanitarias debidas a la pandemia por covid-19 impidieron a los y las niñas asistir a los LIE, como consecuencia en esos años se tuvo muy poca producción de proyectos.

Se desarrollaron tres casos de estudio, el primero consideró 50 proyectos seleccionados aleatoriamente, el segundo 500 proyectos seleccionados

aleatoriamente y el tercero consideró la totalidad de los datos (proyectos) almacenados en GECO, como se muestra en la Tabla 1:

Tabla 1: Cantidad total de proyectos en el servidor GECO.

Grado	2018	2019	2020	2021	2022
Tercer		137	5		1
Cuarto	164	2,484	3		
Quinto	10	489	2	1	
Sexto			3	4	1
Total	174	3,110	13	5	2

b) Diseño de la investigación:

Se aplicó una metodología investigación-acción mediante la cual se desarrollaron propuestas de diseño y de algoritmos de forma colaborativa e iterativa con el equipo de investigación. Se iteró respecto al abordaje del problema, así como a los conjuntos de datos aplicables. Las iteraciones incluyeron las siguientes etapas:

- Recopilación de requerimientos y datos

  El objetivo de esta etapa fue recopilar y refinar los requerimientos y datos de la investigación. Se llevaron a cabo sesiones de trabajo semanales, quincenales o mensuales según el avance en la recopilación de requerimientos. En esta etapa participaron personas docentes, investigadores del Área de Investigación y Evaluación de la FOD, la Dirección del Área Académica de la FOD, investigadores de la Unidad Académica de Estadística de la FOD, consultores expertos (Dr. Alberto Cañas Collado) con quienes se obtuvo información referente al funcionamiento de los laboratorios de informática educativa, al programa LIE++, al tipo de proyectos que desarrollan los estudiantes y al enfoque de enseñanza-aprendizaje constructivista. Asimismo, con el personal de informática de la FOD se coordinó el acceso a

los datos, principalmente a los productos programados por los estudiantes, almacenados en el servidor GECO.

- Análisis de datos

  Se llevó a cabo la revisión y limpieza de la base de datos de ejercicios almacenados en el servidor GECO. Se hizo un primer análisis de los datos para determinar su potencial, características y variables a utilizar para la evaluación del PC. El análisis implicó el desarrollo de un algoritmo de generación de estadísticas generales. Una vez conocidas las características de los datos se procedió a identificar los tipos de patrones que podrían ser obtenidos automáticamente a través del análisis de código. Se procedió a elaborar el primer diseño de los métodos y técnicas de análisis de código y del "framework" para la evaluación automática del código fuente. En cada iteración del proyecto se trabajó con conjuntos de datos seleccionados, así como con la totalidad de los datos con el propósito de ir afinando el diseño del método y los algoritmos.

- Diseño de un "framework" para la evaluación automática de código fuente de los ejercicios de programación

  De forma iterativa, se llevó a cabo el diseño del "framework", el cual arquitectónicamente incluye los componentes necesarios para traducir el código fuente escrito de diferentes lenguajes de programación, a un código genérico, utilizando como base los resultados la del proyecto de investigación previo AVIB. Para este proyecto se incluyó particularmente traducción de código escrito en el lenguaje Scratch (versiones 2.0 y 3.0) a un árbol de sintaxis abstracta (AST) para mapearlo en un árbol general de sintaxis abstracta (GAST). Una vez las instrucciones de código fueron traducidas al GAST, otros

componentes se encargarían de hacer el análisis de patrones de código fuente y el cálculo de métricas. Scratch es el lenguaje más utilizado en los laboratorios de informática educativa, es un lenguaje de programación visual desarrollado por el Media Lab del Instituto Tecnológico de Massachussets (MIT). El "framework" diseñado incluye también componentes para editar de exámenes, así como los respectivos componentes para llevar a cabo su evaluación.

- Diseño de métodos y técnicas de análisis de código

  Se identificaron las gramáticas de los lenguajes de programación y se desarrollaron algoritmos para asociar las instrucciones particulares de AST a las correspondientes en el GAST. El método para efectuar el análisis sintáctico toma como base el GAST.

  Se estableció una escala de 0 a 3 de puntuación para identificar el nivel de desarrollo de habilidades de pensamiento computacional, en donde 0 indica la falta de una habilidad, 1 indica la presencia básica de la habilidad, 2 indica una presencia intermedia y 3 una presencia eficiente.

  Se diseñó un método de análisis sintáctico y la aplicación de la escala para determinar aspectos tales como habilidades en la programación del flujo de control, la representación de datos, modularidad, los errores comunes que cometen los estudiantes, el nivel de dominio de dichas habilidades de cada alumno (con base en la escala previamente definida), la forma en que los estudiantes utilizan los operadores aritméticos, lógicos, de comparación, y las estructuras de control "*if*", "*while*", "*do while*" y "*for*". Con colaboración del equipo de investigación TEC-FOD se determinó el alcance de la investigación, considerando las características de los datos en el repositorio GECO.

Se diseñó un método de clasificación, basado en algoritmos de “clustering”, para analizar el uso que los estudiantes dan a las estructuras de control y el nivel de modularización que utilizan al resolver los ejercicios.

- Desarrollo de casos

Primer caso:

El primer caso tuvo como objetivo determinar si el análisis de elementos sintácticos permitía identificar el nivel de dominio de habilidades computacionales. Se hizo una selección aleatoria de 50 proyectos de los estudiantes, de diferentes niveles de primaria, se aplicó el método de análisis sintáctico y se llevó a cabo el cálculo de estadísticas y de resultados basados en la escala predefinida. Como resultado fue posible determinar, a partir del análisis sintáctico de código fuente, el nivel de domino de habilidades tales como el grado de modularidad de las soluciones y los errores comunes de programación.

Segundo caso:

El segundo caso tuvo como objetivo determinar si a partir de la clasificación de los algoritmos de clustering se pueden determinar niveles de dominio de habilidades de PC, considerando metadatos tales como centro educativo, región, profesor. Se hizo una selección aleatoria de 500 proyectos de diferentes niveles, se aplicaron los algoritmos de análisis de clustering, se hizo una clasificación de los estudiantes de acuerdo con el uso de estructuras de programación que los estudiantes utilizaron en sus soluciones a los ejercicios. Se determinó la utilidad de los algoritmos de clasificación.

Tercer caso:

Finalmente se desarrolló un caso de estudio con los datos suministrados de los años 2018, 2019, 2020, 2021 y 2022. Se aplicaron los algoritmos de

análisis sintáctico y de clustering para determinar niveles de PC en los estudiantes, por nivel (grado), centro educativo, provincia, región, proyectos desarrollados y utilización de estructuras de control *if*, *for*, *while*, así como análisis de la *complejidad ciclomática*. Los resultados fueron visualizados mediante la herramienta Tableau y compartidos y discutidos con funcionarios de la Dirección Académica de la FOD.

- Divulgación de resultados

  Los resultados obtenidos producto de la investigación se han plasmado en seis artículos científicos, cuatro de ellos han sido a la fecha presentados en conferencias y publicados, todos con indexación Scopus y accesibles mediante plataformas tales como IEEE Xplore. Los otros dos artículos se estarán sometiendo a consideración de una revista y de una conferencia a partir del mes de febrero, 2024.

7. Resultados

En esta sección se presentan los principales resultados de la investigación. Se diseñó un framework para la evaluación automática de código fuente, se diseñó un método y algoritmos para el análisis de código fuente con el objetivo de determinar habilidades de pensamiento computacional a partir de productos programados (código fuente) desarrollado por estudiantes del Programa de Informática Educativa-PRONIE durante el período comprendido del año 2018 al 2022, se llevaron a cabo varios casos de estudio, los principales resultados han sido publicados en artículos científicos. Además, dentro del marco de este proyecto de investigación, se desarrolló una tesis de Maestría.

a) Framework para la evaluación automática del código fuente

En la Figura 1 se ilustra el framework para la evaluación automática del código fuente de ejercicios de programación. La parte superior de la figura presenta los componentes del motor de transformación y análisis, el cual toma los datos del repositorio GECO, es decir, programas escritos en Scratch SB2 y SB3, que a través de los componentes AST específico y GAST se traducen a un lenguaje genérico, con lo cual se logra el análisis de código independientemente de la versión de Scratch utilizada. Cabe mencionar que en la figura el foco de atención se centra en Scratch, pero que, producto de un proyecto de investigación anterior (AVIB) se tienen componentes para la transformación de código en otros lenguajes de programación al GAST (por ejemplo, Java).

La parte superior derecha de la figura contiene los componentes del framework para hacer el análisis de código, con independencia del lenguaje de programación. La parte inferior de la figura (bajo la línea azul) describe la arquitectura para editar ejercicios en el lenguaje de programación LIE++ (diseñado específicamente para la FOD) y evaluarlos.

El framework es producto de un diseño iterativo y es valioso para estructurar los procesos para el análisis de PC.

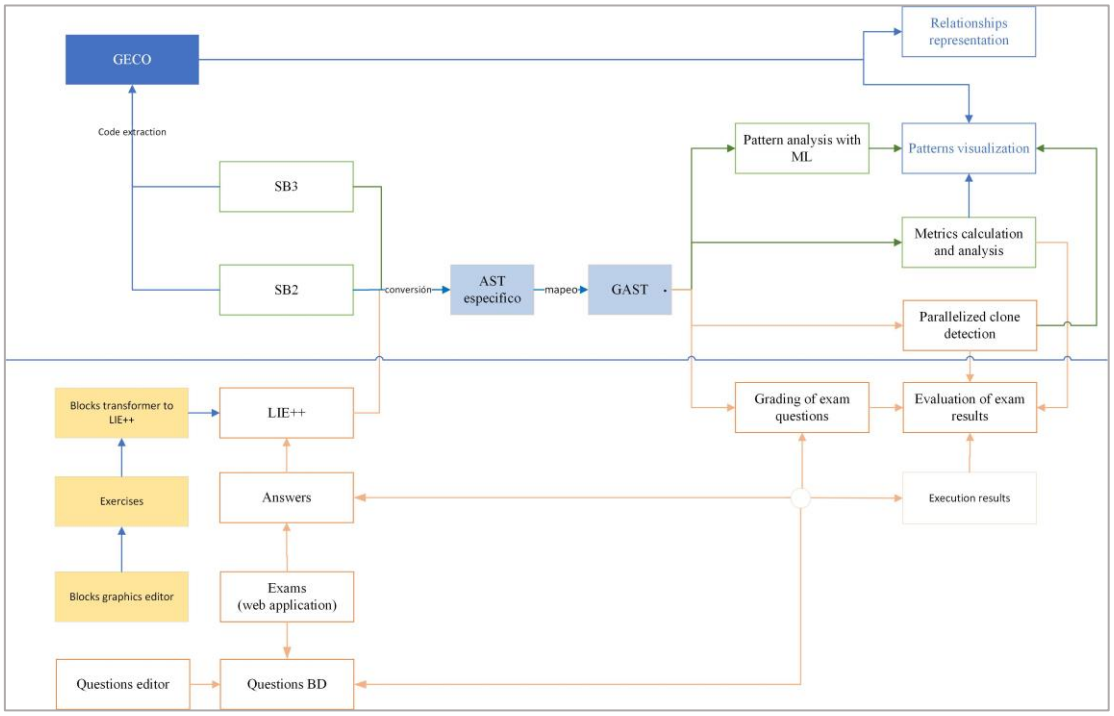

Figura 1: Framework para el análisis de código fuente para determinar habilidades de PC y herramientas de apoyo para la evaluación de ejercicios en LIE++.

b) Algoritmos de análisis de rasgos de habilidades de pensamiento computacional:

Las habilidades de PC evaluadas, considerando las características de los datos en el repositorio GECO son las siguientes:

ABSTRACCIÓN: Representa el nivel de abstracción y descomposición del problema de la solución de acuerdo al nivel de subdivisión del código en pequeños métodos y no en uno solo. El valor máximo es de 3, donde 0 es que no presenta esta habilidad y 3 que lo hace de manera proeficiente. Se evalúa con la rúbrica que se presenta en la Tabla 2.

Tabla 2: Rúbrica de evaluación de Abstracción

Evaluación	Descripción
No presenta la habilidad (0)	Utiliza solamente 1 sprite y una pila de código.
Presenta la habilidad de forma básica (1)	Utiliza solamente 1 sprite pero 2 o más pilas de código.
Presenta la habilidad de forma intermedia (2)	Utiliza 2 o más sprites y al menos uno de ellos con 2 o más pilas de código.
Presenta la habilidad de forma eficiente (3)	Utiliza 2 o más sprites y en al menos uno define sus propios procedimientos y los utiliza.

REPRESENTACIÓN DE DATOS: Representa la capacidad de representar los datos con variables o listas. El valor máximo es de 3, donde 0 es que no presenta esta habilidad y 3 que lo hace de manera proeficiente. Se evalúa con la rúbrica de la Tabla 3.

Tabla 3: Rúbrica de evaluación de Representación de Datos

Evaluación	Descripción
No presenta la habilidad (0)	No crea listas o variables y si las crea no se utilizan.
Presenta la habilidad de forma básica (1)	Crea 1 variable y es utilizada.
Presenta la habilidad de forma intermedia (2)	Crea 2 o más variables y las utiliza.
Presenta la habilidad de forma eficiente (3)	Crea listas y las utiliza.

FLUJO DE CONTROL: Representa la forma en que se tienen bifurcaciones del código para el control de casos. Se calcula utilizando el promedio de la complejidad ciclomática (CYCLO) total entre todos los métodos del código. El valor máximo es de 3, donde 0 es que no presenta esta habilidad y 3 que lo hace de manera proeficiente. La Tabla 4 describe la rúbrica para la valoración de flujo de control:

Tabla 4: Rúbrica de evaluación de Flujo de Control

Evaluación	Descripción
No presenta la habilidad (0)	El promedio de CYCLO entre todas las pilas de código es menor a 1.
Presenta la habilidad de forma básica (1)	El promedio de CYCLO entre todas las pilas de código es mayor a 1 y menor a 2.
Presenta la habilidad de forma intermedia (2)	El promedio de CYCLO entre todas las pilas de código es mayor a 2 y menor a 3.
Presenta la habilidad de forma eficiente (3)	El promedio de CYCLO entre todas las pilas de código es igual o mayor a 3.

PARALELISMO: Representa el nivel de paralelismo de la solución, de acuerdo con la concurrencia de bloques de eventos. El valor máximo es de 3, donde 0 es que no presenta esta habilidad y 3 que lo hace de manera proeficiente. Se evalúa mediante la rúbrica que se presenta en la Tabla 5.

Tabla 5: Rúbrica de evaluación de Paralelismo

Evaluación	**Descripción**
No presenta la habilidad (0)	No presenta ninguna repetición de eventos en el código.
Presenta la habilidad de forma básica (1)	Un evento aparece 2 veces con diferentes conjuntos de instrucciones para cada pila de código.
Presenta la habilidad de forma intermedia (2)	Un evento aparece 3 o más veces con diferentes conjuntos de instrucciones ó dos eventos aparecen 2 veces con diferentes instrucciones por evento
Presenta la habilidad de forma eficiente (3)	Dos o más eventos aparecen 3 o más veces con diferentes conjuntos de instrucciones.

COMPLEJIDAD CICLOMÁTICA: Se calcula haciendo la sumatoria de calcular la complejidad ciclomática de cada método, considera:

1. Cantidad de clases que tiene el código. Se toman como clases cada cantidad de Sprites en el caso de Scratch.

2. Cantidad de métodos en todo el código. Se toman como métodos cada bloque de instrucciones consecutivos en el caso de Scratch.

Los algoritmos para la evaluación de código se encuentran en un repositorio Github: https://bit.ly/3LKq6VI.

c) Resultado del análisis de caso de estudio, aplicando los algoritmos de análisis de código fuente propuestos para determinar rasgos de habilidades de pensamiento computacional.

Las figuras 2 y 3 presentan los principales resultados obtenidos en el último caso de estudio desarrollado. La Figura 2 presenta información general que da contexto en el análisis. En la parte superior izquierda de esa figura se puede observar la cantidad de productos programados para el período del 2018 al 2022, para cada uno de los niveles, de tercer a sexto grado. Nótese que la mayor cantidad de productos programados corresponden al año 2019 (en total 3,110 productos programados), así como a estudiantes de cuarto grado de ese año (2,484 productos programados); esta característica de los datos constituye una restricción para hacer análisis longitudinal del avance de las personas estudiantes en desarrollo de PC; no obstante, representa una cantidad suficiente de datos para llevar a cabo otros análisis.

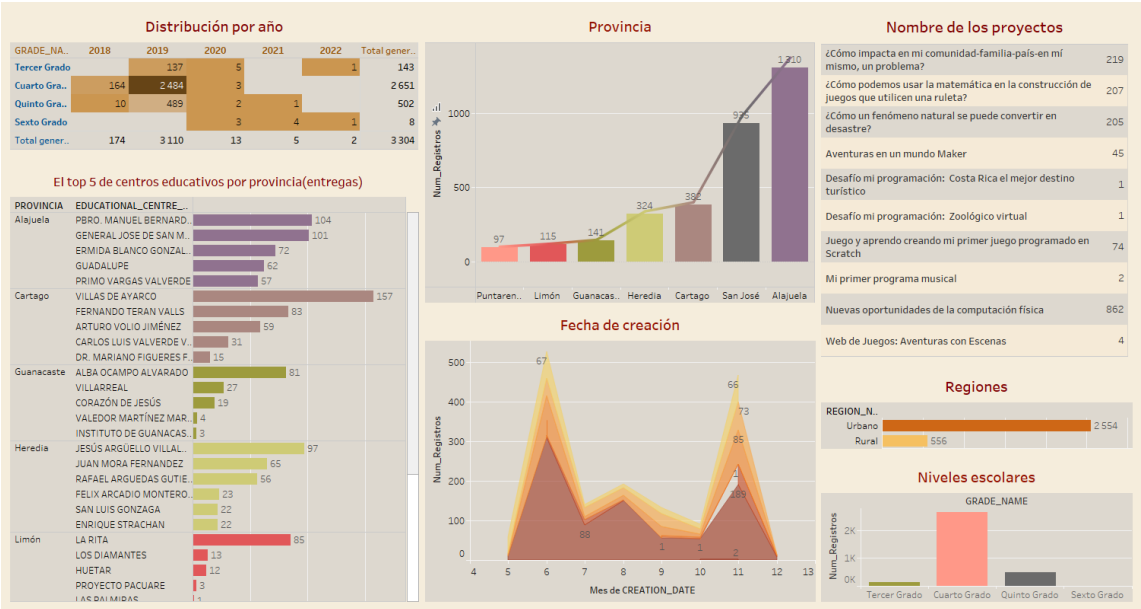

Figura 2: Análisis general de los productos programados.

En la parte inferior izquierda de la Figura 2 se presentan los cinco centros educativos de cada provincia que entregaron la mayor cantidad de productos

programados. Es importante notar la diferencia en cantidad de productos subidos a la plataforma GECO. En la Tabla 6 se detalla la proporción de productos programados en relación a la cantidad de estudiantes beneficiados según los datos proporcionados por la FOD; considerando que los proyectos son frecuentemente desarrollados en grupos de dos personas, a manera de perspectiva se incluye en la tabla una columna que considera esa posibilidad. Casos como los de los centros educativos que reportan más proyectos programados en Cartago, Heredia y Limón, requieren un análisis, por parte de la FOD, de las condiciones que propiciaron esta situación.

Tabla 6: Proporción cantidad de productos programados en el repositorio GECO en relación a la cantidad de estudiantes beneficiados del centro educativo.

Provincia	Lugar	Nombre del Centro Educativo	Cantidad de Productos	Cantidad de Estudiantes Beneficiados	Proporción productos estudiantes	Proporción en caso trabajo en grupos de 2
Alajuela	Palmares	Pbro. Manuel Bernardo Salazar	104	605	17,19%	34,38%
Cartago	Villas de Ayarco	Villas de Ayarco	157	621	25,28%	50,56%
Guanacaste	Liberia	Alba Ocampo Alvarado	81	1,053	7,69%	15,38%
Heredia	San Isidro	Jesús Argüello Villalobos	97	352	27,56%	55,11%
Limón	Guápiles	La Rita	85	1,130	7,52%	15,04%
Puntarenas	Corredores	La Fuente	35	54	64,81%	129,63%
San José	Pavas	Lomas del Río	81	1,206	6,72%	13,43%

Volviendo a la Figura 2, en la parte central de la figura se presenta la cantidad de productos programados entregados por provincia y los meses en que los productos son ingresados desde el centro educativo en la plataforma GECO. Obsérvese los picos en los meses de junio y noviembre, que coinciden con el período previo al término de trimestres, antes de salir a vacaciones. En la parte de la izquierda de la misma figura se listan los proyectos analizados, la cantidad de productos por región (urbana o rural) y cantidad de proyectos por grado.

La Figura 3 presenta resultados específicos del análisis del código de los productos programados. Obsérvese que el proyecto en que se recogieron más soluciones de los estudiantes (productos programados) fue el denominado

"Nuevas oportunidades de la computación física". La provincia donde se registraron más productos programados fue Alajuela (1,310) y menos productos en Puntarenas (97). Los gráficos "**Habilidades de pensamiento computacional**", "**Elementos de sintaxis por nivel**" y "**Pensamiento computacional y complejidad ciclomática**" resumen los resultados del análisis de código fuente aplicando el método de rúbricas establecidas para cada habilidad de pensamiento computacional. En la habilidad paralelismo se obtuvo 1.66, en abstracción 1.70, en representación de datos 1.79. Estos resultados están levemente por encima del promedio, considerando que la rúbrica está en el rango de 0 a 3, siendo representación de datos la habilidad que se observa con mayor desarrollo. En el gráfico de "**Elementos de sintaxis por nivel**", la línea rosada representa 4to grado, la morada 5to grado y la verde 3er grado; obsérvese la tendencia de uso de las estructuras de control en todos los grados y la particularidad de que el rendimiento es, en general, mejor para los estudiantes de cuarto grado.

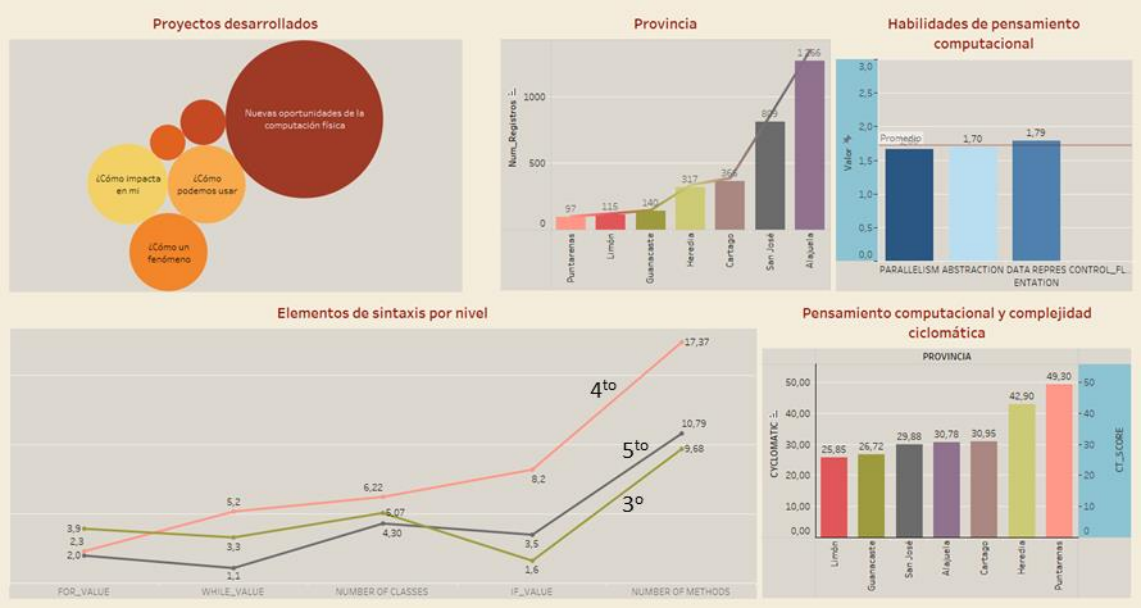

Figura 3: Análisis por proyectos.

Como resultado del análisis de código fuente se obtuvo que es posible obtener y valorar mediante una rúbrica rasgos de habilidades de pensamiento computacional tales como paralelismo, flujo de control, abstracción y representación de datos. La interpretación de los resultados obtenidos en el caso de estudio que se muestra en las figuras anteriores conlleva un análisis profundo de las circunstancias que rodearon cada proyecto y para hacer ese análisis es necesario contar con los antecedentes y la memoria histórica del proceso llevado a cabo en la FOD, para lo cual la participación de las personas funcionarias de esa institución es elemento clave. Esa interpretación será valiosa para definir estrategias y acciones para el desarrollo de ejercicios que promuevan el desarrollo de las habilidades de PC en las personas estudiantes.

d) Resumen de productos resultantes de la investigación

- Artículos científicos. Los aprendizajes obtenidos a lo largo de la investigación se han plasmado en artículos científicos.

  Cuatro artículos publicados son los siguientes:

1) A strategy to assess computational thinking.

   A. Gonzalez-Torres, L. Sancho-Chavarria, M. Zuniga-Cespedes, J. Monge-Fallas and J. NavasSu, "A strategy to assess computational thinking," *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2021, pp. 1037-1042, doi: 0.1109/CSCI54926.2021.00222.

   Este artículo fue aceptado en la International *Conference on Computational Science and Computational Intelligence (CSCI'21)*, llevada a cabo del 15 al 17 de diciembre de 2021. El artículo fue presentado por el Dr. Antonio González Torres. El artículo está accesible en IEEE Explore, que cuenta con indexación Scopus.

2) A method for assessing computational thinking in students using source code analysis.

S. Pacheco-Portuguez, A. Gonzalez-Torres., L. Sancho-Chavarria, I Trejos-Zelaya, J. Monge-Fallas, J. Navas-Su, A. J. Cañas, A. Rodríguez, C. Angulo-Chinchilla. "," *2022 International Conference on Advanced Learning* A method for assessing computational thinking in students using source code analysis *Technologies (ICALT)*, 2022, pp. 144-146, doi: 10.1109/ICALT55010.2022.00050.

Este artículo fue aceptado en la *2022 International Conference on Advanced Learning Technologies (ICALT)*, llevada a cabo del 1 al 4 de julio de 2022 en Bucarest, Rumania de forma híbrida (presencial y virtual). El artículo fue presentado por el Dr. Antonio González Torres de forma virtual. El artículo está accesible en IEEE Explore, que cuenta con indexación Scopus.

3) Automatic Assessment of Programming Exercises Using Syntactic Analysis

E. Ramírez-Trejos, A. González-Torres, L. Sancho-Chavarría, J. Navas-Sú, C. Garita and J. Monge-Fallas, "Automatic Assessment of Programming Exercises Using Syntactic Analysis," *2023 42nd IEEE International Conference of the Chilean Computer Science Society (SCCC)*, Concepcion, Chile, 2023, pp. 1-7, doi: 10.1109/SCCC59417.2023.10315732.

El artículo fue aceptado en el XXIII Congreso Chileno de TICs para la Educación. El artículo fue presentado en el congreso por la Dra. Lilliana Sancho el día 23 de noviembre, 2022. El artículo está accesible en IEEE Explore, que cuenta con indexación Scopus.

4) Towards the Assessment of Basic Computational Thinking Skills Using Syntactic Analysis Techniques

A. González-Torres, E. Ramírez-Trejos, L. Sancho-Chavarría, J. Navas-Sú, C. Garita and J. Monge-Fallas, "Towards the Assessment of Basic Computational Thinking Skills Using Syntactic Analysis Techniques," *2023 2023 Congress in Computer Science, Computer Engineering & Applied Computing*, 2023, pp. 1090-1095, doi: I 10.1109/CSCE60160.2023.00181.

El artículo fue aceptado en la conferencia 2023 *Congress in Computer Science, Computer Engineering & Applied Computing*, 2023 y presentado por el Dr. Antonio González Torres. El artículo está accesible en IEEE Explore, que cuenta con indexación Scopus.

Los siguientes dos artículos elaborados que se someterán a publicación en el 2024 son los siguientes:

5) Artículo: Fundamentos y Perspectivas del Pensamiento Computacional: Un Análisis Integral para la Investigación Futura.

Enviado a la Revista Tecnología en Marcha

Autores: Jorge Monge-Fallas, Lilliana Sancho-Chavarría, César Garita, Antonio González-Torres, Ignacio Trejos-Zelaya.

6) Artículo: Recognition of programming patterns to support the assessment of computational thinking (Título tentativo)

Se enviará a conferencia en el I semestre, 2024.

Autores: Steven Pacheco-Portuguez, Antonio González-Torres, Lilliana Sancho-Chavarría, Ignacio Trejos-Zelaya Jorge Monge-Fallas.

e) Otros informes:

Informes técnicos (reportados con la entrega de los informes de avance a lo largo de la duración proyecto)

- Informe técnico estudio tipo “survey” sobre pensamiento computacional.
- Informe técnico del análisis de los requerimientos de profesores y administradores educativos que intervienen en el proceso de enseñanza de PC.
- Informe técnico con el análisis de las herramientas y métodos disponibles para con el detalle de sus características y posibilidades de reutilización.

- Conjunto de métodos para el análisis avanzado de código, programados y disponibles en repositorio Github.
- Framework para el análisis de ejercicios de programación y la evaluación del cumplimiento de objetivos de aprendizaje.
- Caso de estudio, cuyos resultados se discutieron en el artículo publicado titulado "A method por assessing computational thinking in students using source code analysis".

f) Trabajos finales de graduación

- Tesis de la Maestría en Computación con énfasis en Ciencias de la Computación, del estudiante Steven Pacheco Portugués, titulada "Reconocimiento de patrones de programación para apoyar la evaluación de pensamiento computacional", dirigida por los profesores Dr. Antonio González Torres y el MSc. Ignacio Trejos Zelaya. La tesis se encuentra en el repositorio institucional, accesible por medio de los servicios bibliotecarios del TEC.
- Proyecto final de graduación, carrera Ingeniería en Computadores, del estudiante Ellioth Ramírez, trabajo titulado "Evaluación Automática de Código de Exámenes con base en Pensamiento Computacional", dirigido por el Dr. Antonio González Torres.

8. Discusión y Conclusiones

Con respecto a los objetivos planteados en la propuesta original del proyecto, éstos se lograron en gran medida ya que se definió un método de análisis avanzado de código fuente para comprender los rasgos de habilidades de PC al analizar las características del código fuente de diferentes tipos de ejercicios generados por estudiantes de educación general básica y planteados en casos de estudio. El método propuesto presenta la capacidad para evaluar aspectos fundamentales de la composición estructural del código fuente y, a partir de allí, determinar rasgos de PC. Para tal efecto se diseñó un "framework" para la

evaluación automática de código fuente de ejercicios de programación y se desarrollaron algoritmos para su análisis. El método y los algoritmos fueron diseñados y probados de forma iterativa, y aplicados a través de tres casos de estudio. El primer caso de estudio permitió probar y refinar los algoritmos para el análisis de PC, así como conocer los datos. El segundo caso de estudio permitió obtener una clasificación de los estudiantes de acuerdo con el uso de estructuras de programación que los estudiantes utilizaron en sus soluciones a los ejercicios. Con el tercer caso de estudio se determinaron rasgos de PC, analizando las estructuras de control *if*, *for*, *while*, y la *complejidad ciclomática,* para un total de 3,302 productos programados, desarrollados por estudiantes en los años 2018, 2019, 2021 y 2022., siendo el año 2019 el año de referencia al contar con la mayor cantidad de productos programados (3,110).

Se publicaron cuatro artículos científicos, con indexación Scopus. Se escribieron otros dos artículos que están pendientes de publicación, uno de ellos se envió a consideración de la revista Tecnología en Marcha, el otro se estará enviando a una conferencia durante el primer semestre, 2024.

Además, se lograron los objetivos planteados en la propuesta de ampliación, a saber:

> Objetivo general: Determinar rasgos de habilidades de pensamiento computacional al analizar las características reflejadas en el código fuente de diferentes tipos de ejercicios generados por estudiantes de educación general básica y planteados en un caso de estudio.
> Objetivos específicos:
> a. Desarrollar un caso de estudio, aplicando los algoritmos de análisis de código fuente propuestos, para determinar rasgos de habilidades de pensamiento computacional.
> b. Sistematizar aprendizajes obtenidos a lo largo de la investigación y plasmarlos en artículos científicos.

No obstante, es importante indicar, tal y como se ha reportado previamente en los informes semestrales de avance de este proyecto, que no fue posible desarrollar el "Motor de correlación entre los objetivos de aprendizaje y las habilidades de PC" planteado en el

objetivo OE4, debido a que, una vez iniciado el proyecto, en la etapa de requerimientos, se determinó que a razón del enfoque de aprendizaje constructivista aplicado en los proyectos programados que realizan los estudiantes del PRONIE, los proyectos elaborados por los estudiantes no recibían calificación alguna y la retroalimentación era obtenida durante la experiencia de aprendizaje sin dejar registro alguno en el repositorio GECO. De acuerdo con el enfoque constructivista, los estudiantes aprenden al experimentar los aciertos y los errores y no interesa poner una calificación. La retroalimentación o discusión que se llevaba a cabo en el laboratorio tampoco quedaba registrada. En el repositorio GECO no se registran los enunciados de los ejercicios ni posibles soluciones esperadas por parte del docente, por lo cual no se tiene forma de contrastar los objetivos de aprendizaje con las habilidades de PC identificadas a través del análisis del código fuente.

El proyecto presentó varias limitaciones:

- Limitaciones producto de las restricciones sanitarias impuestas por la pandemia por COVID 19 en los años 2020, 2021 y 2022.

  Inicialmente hubo limitación en cuanto a recolección y vistas presenciales a los laboratorios de informática educativa, pero las actividades se pudieron realizar mediante sesiones virtuales.

  Durante la pandemia, los estudiantes no estuvieron asistiendo a los laboratorios de informática educativa y, en general, en sus casas no contaban con el equipo ni las herramientas tecnológicas para desarrollar los proyectos programados, lo cual impactó fuertemente la cantidad de datos recopilados.

- Limitaciones de datos
  - Según el planteamiento del problema, los datos para esta investigación serían suministrados por la FOD, teniendo como expectativa que la FOD

proporcionara datos de varios años; no obstante, la base de datos que la FOD pudo proporcionar incluía solamente datos del 2018 y 2019.

- Desde el año 2018 los productos programados estaban siendo subidos a la plataforma GECO pero nunca se habían utilizado para generar información. En el proyecto hicimos un análisis inicial de la base de datos, que evidenció algunas inconsistencias en los datos almacenados. Esta situación sirvió de retroalimentación positiva para que la FOD hiciera las correcciones necesarias; sin embargo, fue necesario iterar varias veces hasta lograr un conjunto de datos limpio y confiable.
- Debido a la pandemia por COVID 19, la recolección de datos fue parcialmente suspendida durante 2020, 2021 y 2022, por lo cual se trabajó con menos cantidad de datos que la esperada.
- En el 2023 los centros educativos habían regresado de forma presencial, por lo que se esperaba obtener una mayor cantidad de datos que en los años anteriores de la pandemia. Sin embargo, de forma sorpresiva el MEP no renovó a inicios del 2023 el convenio que había tenido con la FOD durante más de 35 años por lo cual los laboratorios dejaron de funcionar con el PRONIE y a partir de ese momento no fue posible obtener más datos.

- El PRONIE dejó de funcionar
  - Desde el segundo semestre del año 2022, la FOD empezó a experimentar diferencias con el MEP, que provocaron salida de personal clave para el proyecto. La finalización del convenio MEP-FOD a inicios del 2023 provocó despidos masivos que impactaron la contraparte de la FOD. Alrededor de mayo 2023 quedaban solamente un estadístico y la directora académica, con quienes estábamos coordinando un artículo para publicar los resultados de último caso de estudio (figuras 2 y 3 arriba), estos funcionaros estaban haciendo esfuerzos por interpretar la información

generada en ese caso de estudio, pues ya no contaban con la memoria histórica ni con todo el personal necesario. En octubre de ese año la FOD despidió a la directora académica, por lo cual lamentablemente se perdió la posibilidad de interpretación de los resultados desde el punto de vista de la FOD.

El PRONIE era un programa único en el mundo, con una experiencia acumulada de más de 35 años Costa Rica. Si bien fue una limitante para el análisis de datos que el repositorio GECO hubiera sido puesto en producción apenas en el año 2018, hasta donde conocemos, no existe en el mundo otro país que haya contado con los datos y la información generada con el análisis de código llevado a cabo en este proyecto investigación. Es por ello realmente lamentable que los resultados del último caso de estudio no se hubieran podido publicar debido a las dificultades de la FOD como parte del equipo de investigación.

El método, el framework, los algoritmos y la experiencia generada en este proyecto de investigación tiene las bases para, en caso de contar a futuro con datos, poder llevar a cabo análisis profundos que serían valiosos para mejorar los procesos de enseñanza y aprendizaje en el país.

Finalmente, es esencial resaltar que la literatura existente en torno a la conceptualización y evaluación del pensamiento computacional subraya la necesidad de adoptar un enfoque más holístico hacia el mismo. Esto implica el desarrollo de un marco conceptual amplio que no solo englobe la noción de pensamiento computacional, sino que también integre dimensiones cognitivas y sociales. Dicha integración es crucial para facilitar una evaluación comprensiva y multidimensional del pensamiento computacional.

9. Recomendaciones

Los resultados de la investigación indican que es posible desarrollar métodos para la evaluación automática de pensamiento computacional (PC). Siendo considerado el PC valioso para que las personas desarrollen capacidades de pensamiento crítico, de razonamiento y solución de problemas, se recomienda desarrollar investigación futura que considere otros conjuntos de datos que pudieran ser obtenidos a través de actividades específicas en centros de enseñanza de primera, secundaria o universitaria.

10. Agradecimientos

Agradecemos el apoyo de la Vicerrectoría de Investigación y Extensión (VIE) del TEC y al equipo de investigadores de la Fundación Omar Dengo (FOD) que participaron en este proyecto de investigación.

11. Bibliografía

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, *55*(7), 833–835. https://doi.org/10.1093/comjnl/bxs074

Ala-Mutka, K. M. (2005). A Survey of Automated Assessment Approaches for Programming Assignments. *Computer Science Education*, *15*(2), 83–102. https://doi.org/10.1080/08993400500150747

Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. International journal of child-computer interaction, 19, 30-55.

Aman, P. Y., Jon, G., Joke, V., & Fisser. (2017). Computational thinking as an emerging competence domain. *Technical and Vocational Education and Training*, *23*, 1051–1067. https://doi.org/10.1007/978-3-319-41713-4_49

Baker, B. S. (1992). A Program for Identifying Duplicated Code. *Computing Science and Statistics*, *24*, 49–57.

Bull, G., Garofalo, J., & Hguyen, N. R. (2020). Thinking about computational thinking: Origins of computational thinking in educational computing. Journal of Digital Learning in Teacher Education, 36(1), 6-18.

Caiza, J. C., & del Alamo, J. M. (2013). Programming Assignments Automatic Grading: Review of Tools and Implementations. *7th International Technology, Education and Development Conference*, 5691–5700. Retrieved from http://library.iated.org/view/CAIZA2013PRO

Cheang, B., Kurnia, A., Lim, A., & Oon, W. C. (2003). On automated grading of programming assignments in an academic institution. *Computers and Education*, *41*(2), 121–131. https://doi.org/10.1016/S0360-1315(03)00030-7

Chen, H. M., Chen, W. H., Hsueh, N. L., Lee, C. C., & Li, C. H. (2017). ProgEdu - An automatic assessment platform for programming courses. *Proceedings of the 2017 IEEE International Conference on Applied System Innovation: Applied System Innovation for Modern Technology, ICASI 2017*, 173–176. https://doi.org/10.1109/ICASI.2017.7988376

Coleman, D., Ash, D., Lowther, B., & Oman, P. (1994). Using Metrics to Evaluate Software System Maintainability. *Computer*, *27*(8), 44–49. https://doi.org/10.1109/2.303623

Cutumisu, M., Adams, C., & Lu, C. (2019). A Scoping Review of Empirical Research on Recent Computational Thinking Assessments. Journal of Science Education and Technology, 28(6), 651-676.

Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, *4*(4), 355–369. https://doi.org/10.1007/s40692-017-0090-9

Floyd, L. A. (2020, February). Computational Thinking in Teacher Education. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (pp. 1412-1412).

FOD (2018). Propuesta educativa de primer ciclo. Fundación Omar Dengo. San José, Costa Rica.

FOD (2019). Propuesta Educativa LIE++: pensar, crear, programar. PRONIE-MEP-FOD. Fundación Omar Dengo, San José, Costa Rica

Futschek, G. (2006). Algorithmic Thinking: The Key for Understanding Computer Science, 159–168. https://doi.org/10.1007/11915355_15

Halstead, M. H. (2003). Toward a theoretical basis for estimating programming effort. In *ACM '75 Proceedings of the 1975 annual conference* (pp. 222–224). Association for Computing Machinery (ACM).

https://doi.org/10.1145/800181.810326

Kamiya, T., Kusumoto, S., & Inoue, K. (2002). CCFinder- A multilinguistic.pdf, *28*(7), 654–670.

Kemmis, S., & McTaggart, R. (2005). Participatory Action Research: Communicative Action and the Public Sphere. In *The Sage handbook of qualitative research, 3rd ed.* (pp. 559–603). Thousand Oaks, CA: Sage Publications Ltd.

McCabe, T. J. (1976). A Complexity Measure. *IEEE Transactions on Software Engineering*, *SE-2*(4), 308–320. https://doi.org/10.1109/TSE.1976.233837

Morales-Castellanos, H. (2017). Source code analysis on student assignments using machine learning techniques.

Pawelczak, D., Baumann, A., & Schmudde, D. (2015). A new Testing Framework for C-Programming Exercises and Online-Assessments. *The 11th International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'15), Part of WorldComp'15*, 279–285. Retrieved from http://worldcomp-proceedings.com/proc/p2015/FEC2885.pdf

Peteranetz, M. S., Morrow, P. M., & Soh, L. K. (2020, February). Development and Validation of the Computational Thinking Concepts and Skills Test. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (pp. 926-932).

Poženel, M., Fürst, L., & Mahnič, V. (2015). Introduction of the automated assessment of homework assignments in a university-level programming course. *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 - Proceedings*, (May), 761–766. https://doi.org/10.1109/MIPRO.2015.7160373

Román-González, M., Pérez-González, J. C., and Jiménez-Fernández, C. (2015) Test de pensamiento computacional: diseño y psicometría general. III Congreso Internacional sobre Aprendizaje, Innovación y Competitividad (CINAIC 2015), 1–6.

Roy, C. K., Cordy, J. R., & Koschke, R. (2009). Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of Computer Programming*, *74*(7), 470–495. https://doi.org/10.1016/j.scico.2009.02.007

Seehorn, D. et al (2011). CSTA K–12 Computer Science Standards: Revised 2011," ACM.

Sondakh, D. E., Osman, K., & Zainudin, S. (2019, October). Holistic Assessment of Computational Thinking for Undergraduate: Reliability and Convergent Validity. In Proceedings of the 2019 11th International Conference on Education Technology and Computers (pp. 241-245).

Sondakh, D. E., Osman, K., & Zainudin, S. (2020). A Proposal for holistic assessment of computational thinking for undergraduate: Content validity. European Journal of Educational Research, 9(1), 33-50.

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. Computers & Education, 148, 103798.

Wagner, S., Abdulkhaleq, A., Bogicevic, I., Ostberg, J.-P., & Ramadani, J. (2016). How are functionally similar code clones syntactically different? An empirical study and a benchmark. *PeerJ Computer Science*, *2*, e49. https://doi.org/10.7717/peerj-cs.49

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical transactions. Series A, Mathematical, physical, and engineering sciences. 366. 3717-25. 10.1098/rsta.2008.0118.

Zúñiga-Céspedes, M. (2018). Enseñanza y aprendizaje de programación en Costa Rica: aprendizajes desde la investigación cualitativa. *+Aprendizajes*, *1*(2), 62–65.

Zipitría, S. D. R. (2018, October). Piaget and Computational Thinking. In Proceedings of the 7th Computer Science Education Research Conference (pp. 44-50).

12. Apéndice: artículos publicados