Escritura de texto y producción de código limpio: dos realidades de un mismo proceso en estudiantes de ingeniería

Escrever texto e produzir código limpo: duas realidades do mesmo processo para estudantes de engenharia

Text writing and clean code production: two realities of the same process in engineering students

Cristian Olivares-Rodríguez [*1], **Gabriel Valdés-León** [†2,3], **Martha Vidal-Sepúlveda** [‡1] **y Romina Oyarzún-Yañez** [§4]

[1] *Universidad Austral de Chile, Instituto de informática, Facultad de Ciencias de la Ingeniería, Chile.*
[2] *Universidad de Las Palmas de Gran Canaria, Facultad de Educación, España.*
[3] *Universidad Bernardo O'Higgins, Centro de Investigación en Educación (CIE), Chile.*
[4] *Universidad Andres Bello, Chile.*

Resumen

La calidad de los productos de software depende en gran medida de la capacidad de los desarrolladores de generar código limpio, puesto que permiten incrementar el ciclo de vida del software. Por ello, resulta crucial mejorar las prácticas pedagógicas de la enseñanza de la programación, en particular, la capacidad de escribir código de calidad por parte de los estudiantes. Sin embargo, en la literatura no se reconocen modelos pedagógicos integrales que guíen el desarrollo de esta capacidad de escritura, tal como se observa en la escritura de textos. El objetivo de este trabajo es relacionar la producción de código limpio con las concepciones sobre el proceso de escritura en estudiantes de Ingeniería Informática. Para ello, se diseñaron tres evaluaciones prácticas que permitieron acompañar el proceso de programación de código limpio de los estudiantes universitarios para, posteriormente, relacionar los resultados con las percepciones sobre la escritura que este grupo posee. Dentro de los principales resultados, destaca la relación entre el rendimiento en las tareas de programación y el año de ingreso, así como la correlación positiva entre la producción de código limpio y las concepciones sobre la escritura. Esto abre un espacio inexplorado de colaboración transdisciplinar que permita avanzar hacia un modelo pedagógico que dirija la enseñanza de la escritura de código limpio que contribuya con productos de software de mayor calidad.

Palabras clave: Programación computacional. Escritura. Educación en Ingeniería.

Resumo

A qualidade dos produtos de *software* depende em grande parte da capacidade dos desenvolvedores de gerar código limpo, uma vez que eles permitem aumentar o ciclo de vida do *software*. Portanto, é crucial melhorar as práticas pedagógicas de ensino de programação, em particular, a capacidade dos estudantes de escrever código de qualidade. Entretanto, a literatura não reconhece modelos pedagógicos abrangentes que orientam o desenvolvimento dessa capacidade de escrita, como observado na redação de textos. Este trabalho relaciona a produção de código limpo com as concepções do processo de escrita em estudantes de Engenharia da Computação. Para esse fim, utilizamos três avaliações práticas para avaliar o processo de programação de código limpo em estudantes universitários, para, posteriormente, relacionar os resultados com as percepções sobre a escrita que esse grupo tem. Os resultados mostram uma relação entre o desempenho nas tarefas de programação e o ano de ingresso, e uma correlação positiva entre a produção de código limpo e as concepções sobre a escrita. Concluímos que nosso estudo abre uma colaboração transdisciplinar. Ele pode avançar em direção a um modelo pedagógico para orientar o ensino da escrita limpa de código que contribui para produtos de *software* de maior qualidade.

Palavras-chave: Programação de computadores. Linguagem escrita. Educação em engenharia.

*Email: cristian.olivares@uach.cl
†Email: gabrielsebastian.valdes@ulpgc.es
‡Email: martha.vidal@uach.cl
§Email: r.amaliaoyarzun@uandresbello.edu

Textolivre
Linguagem e Tecnologia

DOI: 10.1590/1983-3652.2023.41439

Sección:
Artículos

Autor correspondiente:
Cristian Olivares-Rodríguez

Editor de sección:
Danierverlin Pereira
Editor de maquetación:
Leonado Araújo

Recibido el:
6 de octubre de 2022
Aceptado el:
22 de enero de 2023
Publicado el:
17 de marzo de 2023

Abstract

The quality of software products depends to a large extent on the ability of developers to generate clean code since they allow increasing the software life cycle. Therefore, it is crucial to improve the pedagogical practices of teaching programming, in particular, the ability of students to write quality code. However, the literature does not recognize comprehensive pedagogical models that guide the development of this writing ability, as observed in text redaction. This paper relates the production of clean code with the conceptions of the writing process in Computer Engineering students. Thereby, we used three practical assessments to evaluate the process of clean code programming on university students to later relate the results with the perceptions about writing that this group has. Results show a relationship between performance in programming tasks and the cohort and a positive correlation between the production of clean code and conceptions about writing. We conclude that our study opens a transdisciplinary collaboration. It can advance towards a pedagogical model to guide the teaching of clean code writing that contributes to higher quality software products.

Keywords: Computer programming. Written language. Engineering education.

1 Introducción

La calidad del producto de software se sustenta, principalmente, en la capacidad de producir código limpio. Un código limpio contribuye con la escalabilidad, reutilización y optimización de las componentes de software del producto construido. Sin embargo, observamos que el principal desafío es integrar estos aspectos técnicos avanzados con una mayor sistematización pedagógica, contribuyendo con la calidad de los productos desarrollados en contextos universitarios, en concreto, a su estabilidad y continuidad del proceso de negocio soportado por el servicio tecnológico. Esto no debe dejar de lado la complejidad observada en los cursos introductorios de programación (FIGUEIREDO y GARCÍA-PEÑALVO, 2022).

La motivación de este estudio radica en el cambio cultural de las actuales fábricas de software, las que se encuentran en un tránsito desde la centralización de las tareas de aseguramiento de la calidad en metodologías rígidas a la distribución de las responsabilidades de la producción desde etapas tempranas en metodologías ágiles. Para ello, es necesario distribuir el trabajo y responsabilidades entre los miembros del equipo. De esta manera, cobra relevancia la formación en competencias de trabajo en equipo, colaboración y en la producción de código limpio que permitan la integración continua y una fácil lectura por cada uno de los desarrolladores que forman o formarán parte de un equipo.

El objetivo de esta investigación es relacionar la producción de código limpio con las concepciones sobre el proceso de escritura en estudiantes de Ingeniería informática. De este objetivo, se desprende nuestro interés por poner en diálogo el mundo de la escritura "tradicional" con el de la producción de código informático, lo que nos lleva a indagar en torno a una eventual correlación entre estas dos variables. El valor de esto radica en que, de corroborarse un vínculo entre ellas, sería posible revisitar las décadas de investigación en cuanto a la didáctica de la escritura con el fin de aplicar estos avances en el ámbito de la enseñanza de la programación.

La principal contribución de este estudio es la habilitación del camino hacia nuevas prácticas pedagógicas para la enseñanza de la producción de código limpio en estudiantes universitarios, dada la positiva relación observada entre la producción de código limpio y las percepciones de escritura. Asimismo, en el diseño de nuevas prácticas es imprescindible integrar las habilidades y percepciones de escritura/lectura de textos.

2 Trabajos relacionados

La producción de código limpio implica la reducción y/o eliminación de los *code smells* que emergen de las primeras versiones del producto, ya que son los principales responsables de la reducción de la calidad del producto de software. Estos *code smells* están directamente vinculados con los posibles fallos futuros del software dificultando la capacidad de mantención del producto, lo que se traduce en limitaciones en la calidad del servicio del negocio en el cual se inserta el desarrollo.

Los *code smells* son síntomas observables o anomalías que emergen durante el proceso de escritura del código y, por tanto, limitan la evolución del producto de software debido a que complejizan su

lectura. Se ha propuesto este término para referirse a patrones que contienen problemas estructurales observables en el código que deberían ser mejorados (FOWLER, 2018). Desde la conceptualización de Fowler y sus colegas, se han propuesto diferentes listas de identificación y organización de *code smells* en los productos de software. Se han identificado antipatrones de desarrollo, los cuales integran elementos arquitectónicos, de desarrollo y de gestión que se traducen en consecuencias negativas para el producto (BROWN; MALVEAU y col., 1998). Asimismo, se han identificado *code smells* con el propósito de vincularlos con técnicas de *refactoring* pertinentes para dichos síntomas (FOWLER, 2018; LACERDA y col., 2020). Estas listas han sido complementadas con nuevas categorías de problemas que han sido capturados desde la práctica profesional (GUPTA; KAPUR y KUMAR, 2016) o extendidas a antipatrones presentes en productos basados en patrones arquitectónicos comúnmente utilizados en la industria del software, tal como el modelo MVC (ANICHE y col., 2018) o extendidas a tipos de productos en particular, tales como los desarrollos de productos de *machine learning* (OORT y col., 2021; LIU; JIN y col., 2019).

2.1 La enseñanza de código limpio

La programación es el proceso de resolución de problemas a través de algoritmos computacionales, que corresponden a una secuencia de instrucciones que permiten dar respuesta a una necesidad de información por medio de la implementación sobre un lenguaje de programación apropiado al contexto. Por lo tanto, la enseñanza de la programación debe ser abordada desde la perspectiva del pensamiento computacional (WING, 2008; VOOGT y col., 2015), contemplando el desarrollo de la capacidad de resolución de problemas computacionales, entendida como un abordaje metodológico que busca comprender las necesidades de información en el dominio del problema, extraer la información relevante del dominio por medio de abstracciones en el espacio de solución y establecer una secuencia lógica que resuelva el problema.

El proceso de análisis abstracto debe llegar a ser implementado de manera concreta, por medio de la escritura de código, en un lenguaje de programación. Esto impondrá restricciones a la solución diseñada en función de su paradigma y estructura sintáctica. Mientras que el paradigma de programación establece las reglas de composición de las piezas de código, la sintaxis define las reglas de escritura del lenguaje. Este proceso de comprensión del problema y creación de soluciones interactúa tanto con el pensamiento crítico (BASTÍAS; DÍAZ y RODRÍGUEZ, 2021), en la evaluación del problema; como con el pensamiento creativo, durante la evaluación de las alternativas de solución (EGUILUZ y col., 2017; GROENEVELD y col., 2022). En consecuencia, los programadores deben ser capaces de resolver problemas por medio de la escritura de código o codificación, ajustándose a las reglas del lenguaje sobre el cual se escribe para establecer una solución apropiada y de calidad.

Por lo tanto, los cursos de programación en la educación superior, principalmente en los programas de estudio asociados a las ciencias de la computación, se orientan a desarrollar la capacidad de escribir código que permita resolver problemas simples y se estructuran en torno a dos ejes formativos: la resolución de problemas o nivel abstracto; y la comprensión lenguaje de programación o nivel operativo (IQBAL MALIK y col., 2021). De esta manera, se han propuesto diversos acercamientos pedagógicos que dependen del énfasis con el cual se combinan estos ejes.

En particular, para el desarrollo de las habilidades de programación se han propuesto una serie de prácticas pedagógicas que apuntan a mejorar la calidad del código generado por los estudiantes. Más allá de la selección del lenguaje de programación, la orientación pedagógica ha delineado dos tipos de prácticas: 1) la lectura de código y 2) la escritura de código; tendiendo en consideración que ambas habilidades se encuentran estrechamente relacionadas.

Una de las principales prácticas pedagógicas en la enseñanza de la programación es la escritura de código. Esta se centra en la formación de habilidades técnicas asociadas al lenguaje de programación y la capacidad de los estudiantes para establecer una secuencia algorítmica desde el análisis abstracto. Fincher (1999) describe cuatro acercamientos metodológicos para la enseñanza de la programación enfocados en las capacidades de análisis de los aprendices antes de avanzar en la codificación. Esto se encuentra alineado con las propuestas actuales basadas en la programación por bloques (KYFONIDIS; MOUMOUTZIS y CHRISTODOULAKIS, 2017; WEINTROP, 2019), lo que reduce la complejidad en

la adquisición del lenguaje y se enfoca en la capacidad de abstracción. Sin embargo, la estrategia pedagógica clásica es la de sesiones teóricas seguidas de prácticas, autónomas o guiadas (MURPHY; CRICK y DAVENPORT, 2016). Mientras que los principales lenguajes de programación son Java y, actualmente, Python (BECKER y QUILLE, 2019), lo que puede estar relacionado con sus características de tipado débil. Sin embargo, estas prácticas apuntan a la formación en cursos introductorios, donde los estudiantes deben comprender el proceso de resolución de problemas por medio de algoritmos computacionales.

Mientras que, por otra parte, la lectura de código como práctica pedagógica aborda el fomento de la codificación limpia por medio de la capacidad de análisis de la sintaxis del lenguaje de programación y las semánticas de las soluciones escritas previamente. Por consiguiente, la introducción de *bugs* en códigos ha sido utilizada como práctica pedagógica, reflejando que los estudiantes que reciben este tipo de códigos mal escritos alcanzan mejores resultados de aprendizaje que sus pares (GRIFFIN, 2019). Similarmente, la enseñanza del código limpio desde la lectura está dirigida por la retroalimentación oportuna, la cual comienza por sesiones teórico-prácticas de lectura de código de manera interactiva, sesiones de revisión de tareas en directo con los estudiantes y la evaluación individual y oral de las competencias de codificación limpia (DIETZ y col., 2018). Mientras que otro estudio aborda el diseño de un tutor inteligente que presenta a los estudiantes trozos de código de baja calidad para evaluar su calidad, contribuyendo con la capacidad de producción código limpio (PROKIĆ y col., 2021). Por lo tanto, la capacidad de lectura de códigos influye positivamente en la capacidad de escritura de los educandos. Esta relación trae consigo, de manera implícita, que la capacidad de producción de código limpio se encuentra ligada a la capacidad de lectura de código debido al fomento del proceso reflexivo y analítico que trae consigo la exploración de trozos de código.

2.2 Concepciones sobre el proceso de escritura

Durante los últimos años, los estudios sobre escritura en la educación terciaria han tenido un aumento exponencial (NAVARRO y col., 2016). Mucha tinta se ha vertido en relación con la importancia de acompañar el proceso de alfabetización académica de los estudiantes universitarios con énfasis en los de nuevo ingreso (ROALD y col., 2021), quedando en segundo plano las investigaciones realizadas con alumnos de nivel de posgrado (CALLE-ARANGO y ÁVILA REYES, 2022).

Lo anterior se justifica debido a la importancia que tiene la escritura en la formación universitaria, principalmente en los primeros años, e independiente del área epistémica. Diversos son los estudios que establecen una estrecha relación entre el éxito académico y las competencias comunicativas productivas (ULU, 2019; TURKBEN, 2021) y receptivas (KILNER; COLLIE y CLEMENT, 2019; LIU y READ, 2020), pero, además, su importancia en el crecimiento disciplinar de los aprendientes.

Al respecto, autores como Walton (2020) o Selwyn y Renaud-Assemat (2020) destacan la importancia del dominio de estas habilidades para los estudiantes y profesionales de la ingeniería, pero, al mismo tiempo, dan cuenta de que su desarrollo es aún un desafío pendiente. En palabras de Goldsmith, Willey y Boud (2019, p. 71):

> Much has been written about the importance of getting engineering students to write, but there has been a little investigation of engineering academics' perceptions of writing practices in the curriculum, and the extent to which these practices are visible to their students and to the academics.

La presente investigación se enfoca, precisamente, en el campo de los estudios sobre las concepciones que tienen los aprendientes sobre la escritura, debido a que uno de los ámbitos que resulta más enriquecedor para los actores del proceso educativo es el de las percepciones en torno a la escritura. Esto se debe a que el manejo de esta información permite acompañar de manera efectiva los procesos de enseñanza, pues se ha comprobado que existe una estrecha relación entre las concepciones de la escritura y los resultados de aprendizaje, tal como señalan Miras, Solé y Castells (2013) en el nivel secundario o González, González y González-Ocampo (2020) en el nivel terciario, por nombrar un par de ejemplos.

Al abordar las concepciones en torno a la escritura, distintas son las perspectivas que podemos adoptar. Por ejemplo, algunas investigaciones indagan en cómo conciben los estudiantes la escritura,

vale decir, si valoran su dimensión epistémica o se quedan simplemente con su función reproductiva (GONZÁLEZ; GONZÁLEZ y GONZÁLEZ-OCAMPO, 2020). Por su parte autores como (LONKA y col., 2014) se interesan en un mayor número de aspectos, muchos de ellos relacionados con el proceso de escritura: bloqueos, procrastinación, perfeccionismo, habilidad innata, transformación del conocimiento y productividad. Ahora bien, para los fines de nuestra investigación, hemos adoptado la propuesta de Vine-Jara (2020), pues se enfoca, específicamente, en el proceso de escritura. La autora, sobre la base de los trabajos de Difabio de Anglat (2012) o Rodríguez Hernández y García Valero (2015), aborda las tres instancias del proceso de escritura (antes, durante y después) a través de cinco dimensiones: planificación, textualización, revisión, postergación del inicio de la escritura y actitudes hacia la escritura.

3 Materiales y métodos

El trabajo ha sido diseñado como un estudio transversal, pues la información fue recogida en un único momento. Asimismo, posee carácter descriptivo, ya que busca conocer cómo operan las dos variables consideradas en una situación particular. Por último, posee un alcance correlacional, debido a que entrega un explicación -parcial, por cierto- al fenómeno estudiado.

3.1 Participantes

Participaron de la investigación 28 estudiantes de la asignatura de ingeniería de software de Ingeniería civil en informática. Se seleccionó dicha muestra debido a la estrecha relación entre la investigación y el objetivo principal del curso. Específicamente, esta asignatura tiene como objetivo que los estudiantes sean capaces de desarrollar productos de software usables y de calidad por medio de métodos sistemáticos utilizados en la industria del software. Por lo tanto, es necesario que apliquen las habilidades de resolución de problemas y de escritura de código adquiridas en los primeros años de la carrera de manera guiada. Asimismo, en la asignatura se revisan los siguientes contenidos técnicos: 1) Metodologías de desarrollo de software, 2) Codificación limpia y 3) Calidad de software.

3.2 Protocolo experimental

Para alcanzar el objetivo se planteó un estudio relacional cuasi-experimental en la asignatura de ingeniería de software. En la Figura 1 se presenta el protocolo experimental utilizado en este estudio.

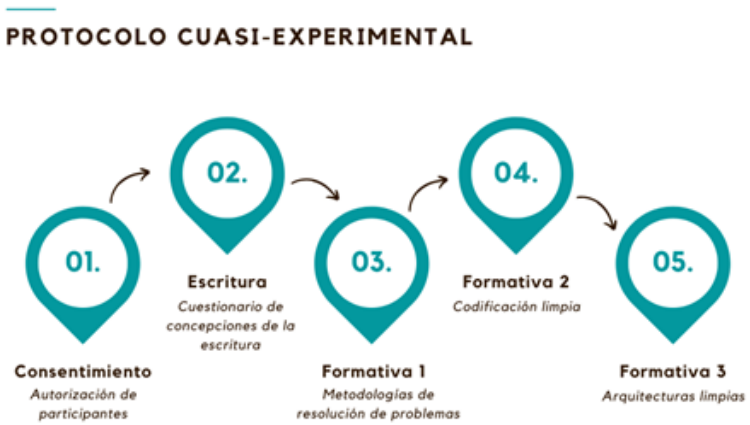

Figura 1. Protocolo cuasi-experimental.
Fuente: Elaboración propia.

3.3 Instrumentos de evaluación del producto de software

Para evaluar la capacidad de desarrollo de software adquiridas por los participantes tanto en la asignatura intervenida como en los primeros años de la carrera, se aplicaron tres evaluaciones prácticas basadas en la resolución de 3 casos de estudios reales.

- **Evaluación 1 "Metodologías de resolución de problemas"**: En la primera actividad evaluativa,

los estudiantes debían establecer los límites del problema a resolver y planificar una estrategia de resolución por medio de un producto de software. Al respecto, era indispensable que establecieran las abstracciones necesarias para su implementación en un entorno de desarrollo web. Esta actividad consideraba:
  - Análisis del problema.
  - Modelado de la solución.
  - Planificación del proyecto.
- **Evaluación 2 "Codificación limpia"**: En la segunda actividad evaluativa, los estudiantes debían establecer un modelado arquitectónico de la solución, considerando la aplicación de patrones de diseño y buenas prácticas de codificación. En esta oportunidad, los estudiantes debían implementar la solución de manera modular en un lenguaje de programación web, considerando lo siguiente:
  - Patrones de diseño.
  - Pruebas unitarias.
  - *Code smells*.
  - Refactorización.
- **Evaluación 3 "Arquitecturas limpias"**: En la última actividad evaluativa, los estudiantes debían realizar una estimación del esfuerzo y costo del desarrollo de un producto de software y, además, implementar uno de los módulos del producto. En esta ocasión, se consideraban las siguientes actividades:
  - Estimación del producto de software.
  - Modelado arquitectónico limpio.
  - Codificación limpia.

3.4 Instrumento de percepción del proceso de escritura

Las percepciones sobre el proceso de escritura fueron recogidas por medio del cuestionario propuesto por Vine-Jara (2020), el que fue diseñado a partir de los trabajos de Difabio de Anglat (2012) y Rodríguez Hernández y García Valero (2015). Este cuestionario de 36 preguntas evalúa cinco dimensiones: planificación, textualización, revisión, postergación del inicio de la escritura y actitudes hacia la escritura. Se optó por este instrumento debido a que recoge información tanto del proceso como de las actitudes relacionadas con la escritura y, además, porque fue diseñado e implementado por la autora para ser aplicado en el contexto chileno, lo que da cuenta de su pertinencia cultural. La autora comparte el instrumento, de manera íntegra, en (VINE-JARA, 2020).

Finalmente, en relación con la fiabilidad del cuestionario, el coeficiente de alfa de Cronbach se observa en la Tabla 1, lo que permite observar que el instrumento posee una alta consistencia interna.

Tabla 1. Fiabilidad del cuestionario por dimensión.

Dimensión	Alfa de Cronbach
Planificación	0,82
Textualización	0,67
Revisión	0,79
Postergación	0,68
Actitudes	0,76

Fuente: Elaboración propia.

4 Resultados

A continuación, se presentan los resultados del estudio exploratorio experimental desarrollado en el curso de Ingeniería de Software. En primer lugar, se presentan los niveles de logro alcanzados por los estudiantes en las evaluaciones formativas de la asignatura. En segundo lugar, se presentan las distribuciones en la concepción del proceso de escritura. Y, finalmente, se presentan las relaciones

entre el desempeño académico en la producción de software y la concepción de la escritura.

4.1 Desempeño académico

Durante el primer mes del periodo lectivo, se analizaron diversas estrategias metodológicas utilizadas en la resolución de problemas por medio de la producción de productos de software. Una de las principales competencias a desarrollar es el análisis del dominio del problema, reconociendo los factores externos e internos de la organización para identificarlo apropiadamente. Esto permite a los estudiantes establecer una estrategia coherente con el contexto del problema, diseñando las acciones que deberán ser llevadas a cabo durante el proceso de producción de software. De esta forma, en la primera evaluación práctica, los estudiantes debían resolver como caso de estudio una licitación pública.

La Figura 2 corresponde a la distribución de calificaciones de los estudiantes en la primera evaluación de la asignatura (eje vertical), organizados por año de ingreso (eje horizontal). En ella se observa una relación entre la cohorte y el desempeño observado en las estrategias diseñadas para resolver los problemas. Así, los estudiantes que ingresaron en el año 2018 y 2019 tienden a alcanzar mejores resultados, con una media en torno al 4.5 sobre 7.0. Mientras que los estudiantes de generaciones anteriores tienen una media en torno al 3.8, considerando un 4.0 la calificación mínima para aprobar la evaluación. En consecuencia, se observa que el desempeño se encuentra estrechamente influenciado por el año de ingreso de los estudiantes, lo que podría asociarse al desempeño general de las nuevas generaciones en el programa de estudios.

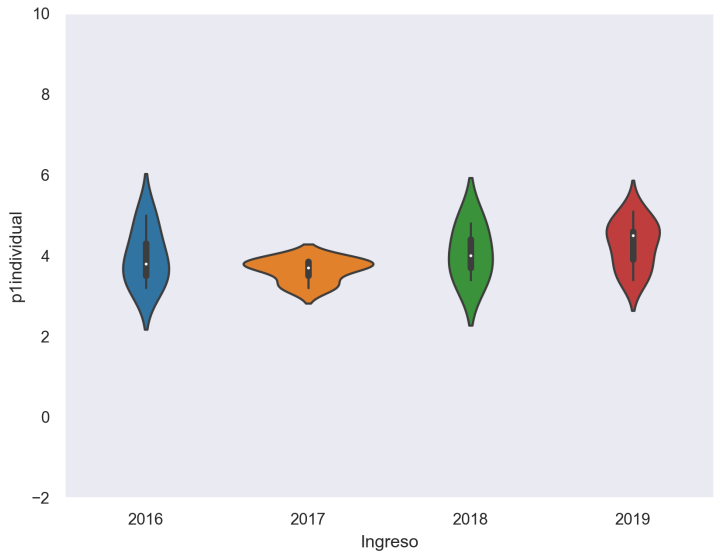

Figura 2. Desempeño en la primera evaluación "Metodologías de resolución de problemas", según año de ingreso.
Fuente: Elaboración propia.

Durante el segundo mes y medio del periodo lectivo, se analizaron diversos enfoques para la producción de código limpio. En primer lugar, se revisaron los principales tipos de patrones de diseño tanto de manera abstracta como aplicados a diferentes lenguajes de programación. Luego, se revisaron estrategias de refactorización dirigidas por la identificación de *code smells* para mejorar la calidad de la producción de código. Finalmente, se revisaron estrategias de diseño e implementación de pruebas para guiar el proceso de mejora de la calidad del código, tanto a nivel de componentes como de manera integral. En la segunda evaluación práctica, los estudiantes debían resolver una nueva licitación pública codificando una de las componentes diseñadas.

La Figura 3 corresponde a la distribución de calificaciones de los estudiantes en la segunda evaluación de la asignatura (eje vertical), organizados por año de ingreso (eje horizontal). Nuevamente se

aprecian diferencias entre generaciones, donde los estudiantes que ingresaron en los años 2018 y 2019 logran un mejor desempeño, con una media de 4.8. Mientras que las generaciones mayores alcanzan calificaciones en torno al 3.3, considerando que la aprobación se logra sobre el 4.0. Por lo tanto, en esta segunda calificación se repite la relación entre la cohorte y el desempeño académico logrado por los participantes.

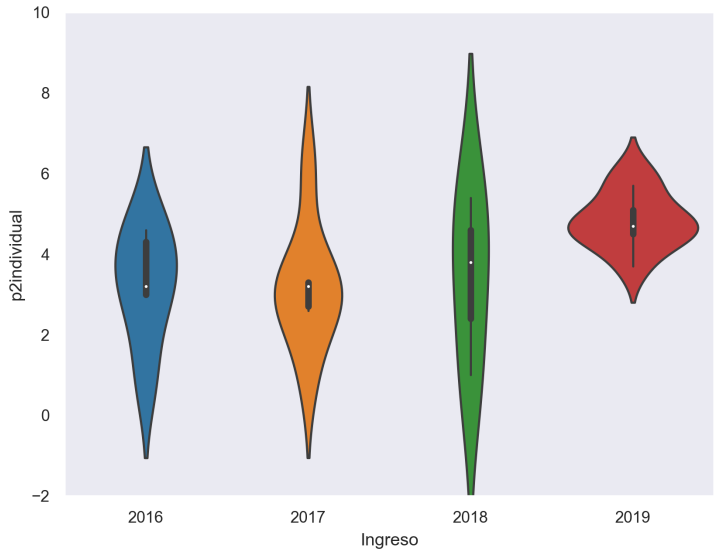

Figura 3. Desempeño en la segunda evaluación "Codificación limpia", según año de ingreso.
Fuente: Elaboración propia.

Durante el último tramo del periodo lectivo, se analizaron diversas estrategias de estimación del tamaño, esfuerzo y costo del proceso de producción de software, las que dependen del paradigma de los lenguajes de programación, el contexto del problema y la arquitectura seleccionada. Para ello, se revisaron modelos de estimación de proyectos de software, desde los más simples hasta los más complejos. Además, se revisaron las implicancias que tienen las arquitecturas limpias tanto en la producción como en la mantención de productos de software. En la tercera evaluación práctica, los estudiantes debían resolver una nueva licitación pública, codificando una de las componentes de la arquitectura limpia diseñada y utilizada para la estimación del producto de software.

La Figura 4 corresponde a la distribución de calificaciones de los estudiantes en la segunda evaluación de la asignatura (eje vertical), organizados por año de ingreso (eje horizontal). Nuevamente se observa la relación entre la cohorte de los estudiantes y el desempeño académico, donde los estudiantes de las nuevas cohortes alcanzan calificaciones en torno al 5.2. Mientras que las cohortes anteriores tienden a obtener calificaciones menores en torno al 3.3, considerando que la aprobación se alcanza con un 4.0. Por lo tanto, en esta tercera calificación se reafirma la relación entre la cohorte y el desempeño académico logrado por los participantes.

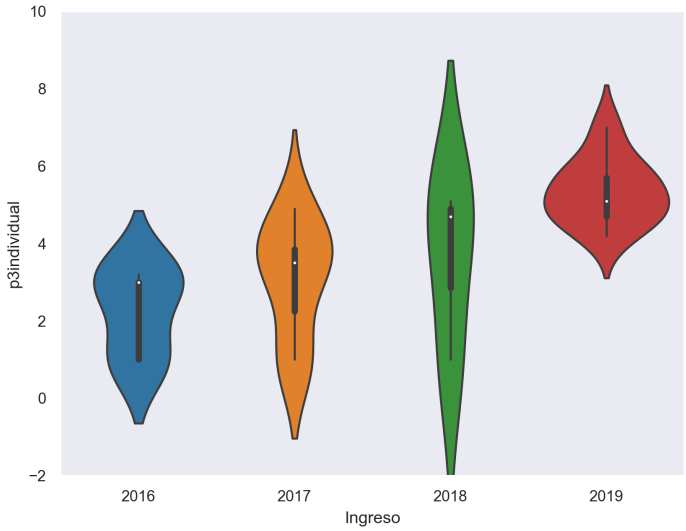

Figura 4. Desempeño en la tercera evaluación "Arquitecturas limpias", según año de ingreso.
Fuente: Elaboración propia.

4.2 Concepción de la escritura

El instrumento utilizado para medir las concepciones sobre la escritura tiene la ventaja de que no solo considera las etapas de este proceso, sino que también entrega información valiosa en torno a las actitudes de los estudiantes en relación con la escritura, y sobre una de las mayores dificultades de los estudiantes al momento de redactar: la dilación del inicio de la escritura. En la Figura 5, se observa que, en términos generales, los resultados tienden a agruparse con una leve tendencia negativa, sobre todo, en las dimensiones de textualización y de postergación. Por su parte, las etapas de planificación y de revisión se presentan como aquellas con mejores resultados.

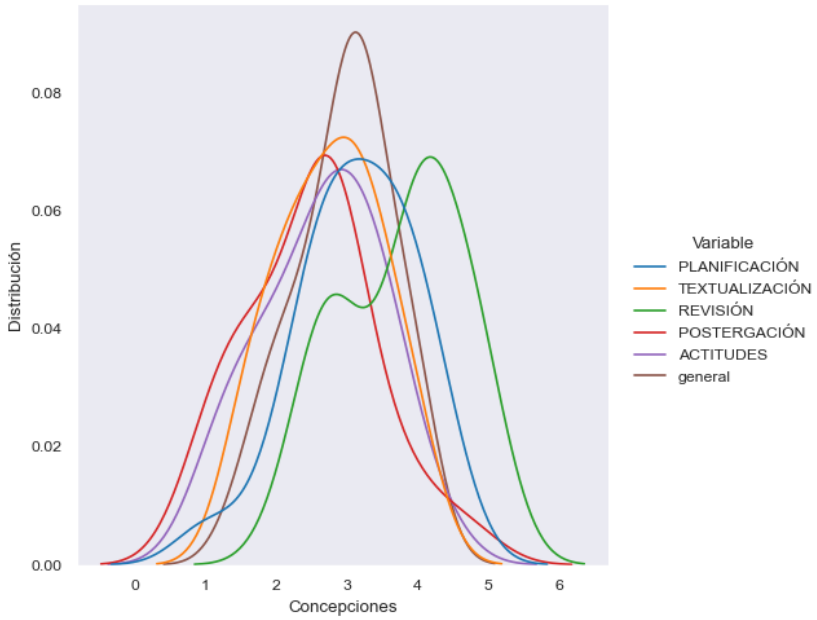

Figura 5. Distribución de concepciones del proceso de escritura, según dimensiones.
Fuente: Elaboración propia.

Tal como señalamos con anterioridad, se observaron resultados interesantes al momento de organizar los datos obtenidos por año de ingreso. En el caso de las percepciones sobre la escritura, en

la Figura 6, se presentan los niveles de percepción de la escritura (eje vertical) en relación al año de ingreso a la universidad (eje horizontal). En ella se vislumbran mejores resultados en la cohorte 2016, seguida de la perteneciente al 2019. Sin embargo, lo verdaderamente llamativo corresponde a la distribución de la cohorte 2017, pues da cuenta de un alto nivel de dispersión en sus resultados. Al comparar este gráfico con el correspondiente al desempeño en codificación limpia (Figura 3), se observa un alto grado de similitud con los datos obtenidos en la segunda evaluación.

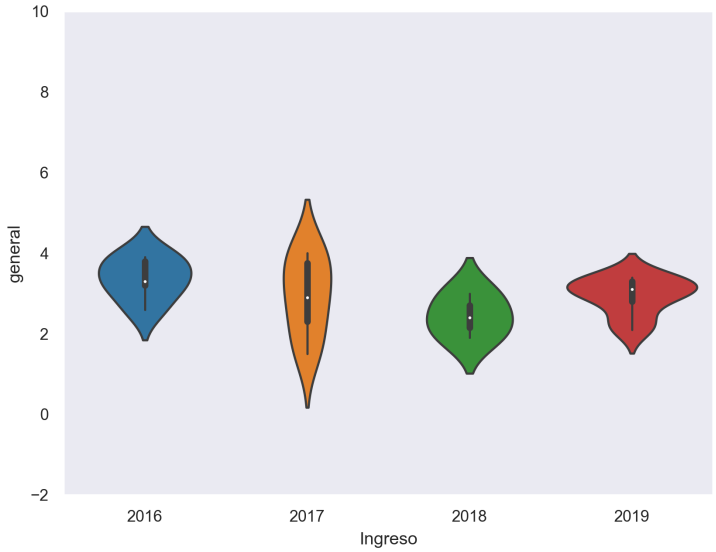

Figura 6. Percepción del proceso de escritura, según año de ingreso.
Fuente: Elaboración propia.

4.3 Escritura de código y de textos

Para evaluar el objetivo de esta investigación, es decir, relacionar las estrategias de producción de código limpio con las concepciones sobre el proceso de escritura en estudiantes de Ingeniería informática se sigue un enfoque relacional basado en la construcción de modelos de predicción de las calificaciones de los estudiantes. En particular, se construyen modelos de regresión lineal que, a partir de las dimensiones de la concepción del proceso de escritura de textos, puedan estimar las calificaciones alcanzadas por los participantes. Debido a que la cohorte ha sido una variable relevante en el desempeño de los estudiantes, se ha decidido incorporar esta dimensión en los modelos predictivos elaborados.

En la Figura 7, se presenta la precisión alcanzada por un modelo de regresión lineal construido a partir de las dimensiones del cuestionario de escritura en conjunto con la variable de cohorte con el propósito de predecir la primera evaluación "Metodologías de resolución de problemas". Se observa que existe una relación positiva, pero no significativa (r=0.53), entre las concepciones de la escritura y la capacidad para establecer una metodología de resolución de problemas apropiada al dominio del caso de estudio abordado por los estudiantes.

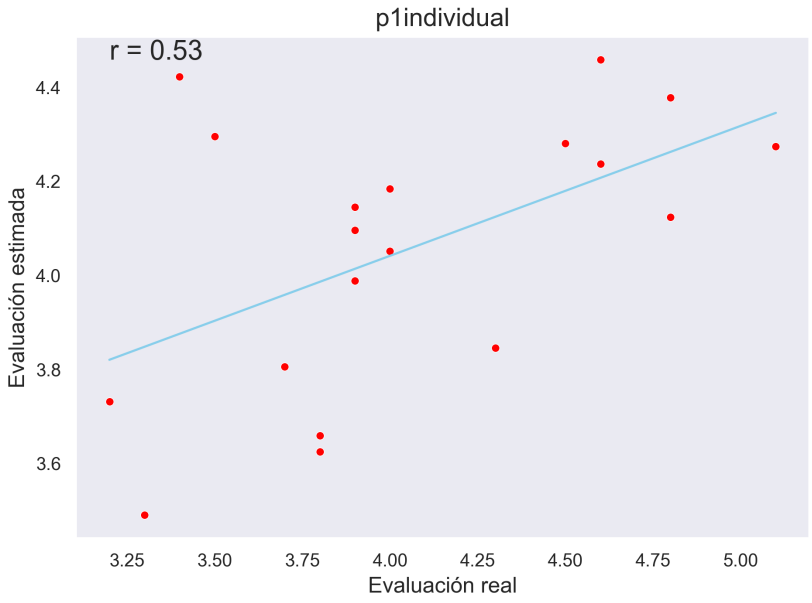

Figura 7. Precisión en la estimación del desempeño en la evaluación 1, según dimensiones de concepción de la escritura.

Fuente: Elaboración propia.

En la Figura 8, se presenta la precisión alcanzada por un modelo de regresión lineal construido a partir de las dimensiones del cuestionario de escritura en conjunto con la variable de cohorte con el fin de predecir los resultados de la segunda evaluación "Codificación limpia". Se observa que existe una relación positiva mayor al primer modelo y significativa (r=0.75) entre las concepciones de la escritura y la capacidad de producir código limpio por medio de la utilización de buenas prácticas de desarrollo apropiadas al dominio del caso de estudio abordado por los estudiantes.

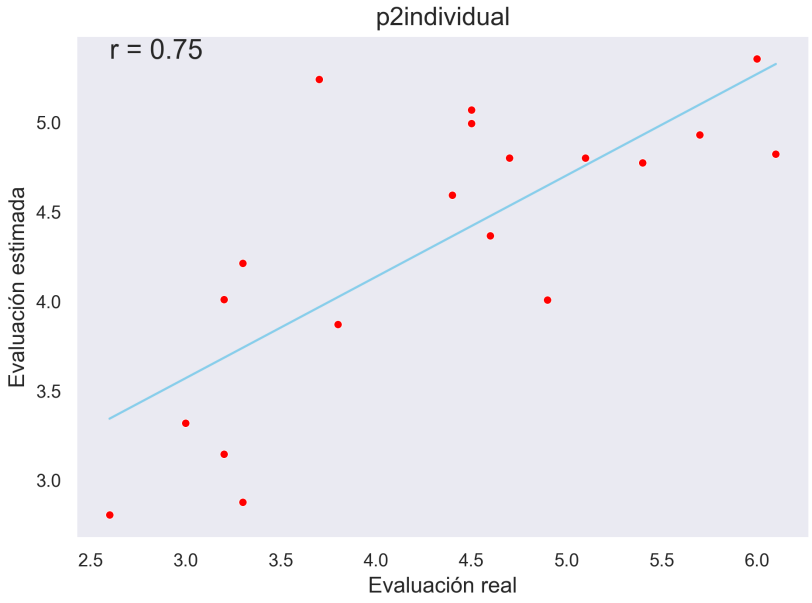

Figura 8. Precisión en la estimación del desempeño en la evaluación 2, según dimensiones de concepción de la escritura.

Fuente: Elaboración propia.

En la Figura 9, se presenta la precisión alcanzada por un modelo de regresión lineal construido a partir de las dimensiones del cuestionario de escritura en conjunto con la variable de cohorte con el fin de predecir los resultados de la tercera evaluación, "Arquitecturas limpias". Se observa que existe una

relación positiva mayor que los dos modelos anteriores y significativa (r=0.87) entre las concepciones de la escritura y la capacidad de producir código limpio por medio de la utilización arquitecturas limpias durante la estimación del esfuerzo de los procesos de desarrollo de software coherentes con el dominio del caso de estudio abordado por los estudiantes.

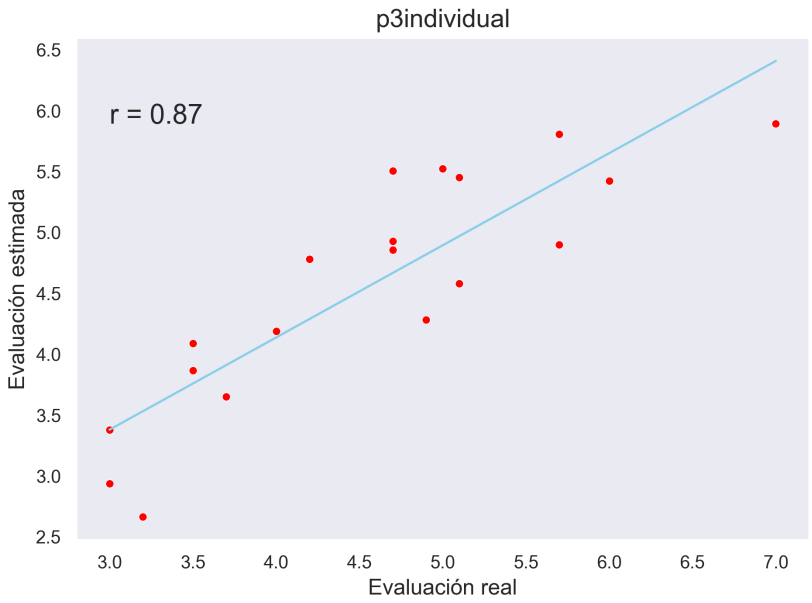

Figura 9. Precisión en la estimación del desempeño en la evaluación 3, según dimensiones de concepción de la escritura.

Fuente: Elaboración propia.

En consecuencia, los modelos elaborados a partir de las concepciones del proceso de escritura de los estudiantes resultan ser más significativos y precisos cuando las evaluaciones se vinculan con la producción de código, tanto en la segunda como tercera evaluación. Mientras que la evaluación de la resolución de problemas presenta una relación positiva con las concepciones de escritura, no se logran altos niveles de precisión por parte de los modelos de predicción. Finalmente, la introducción de la cohorte como variable predictora genera un mayor nivel de precisión a los modelos elaborados.

5 Discusión

Uno de los aspectos más interesantes que surgen desde los resultados, y que está estrechamente vinculado con nuestro objetivo, tiene que ver con cómo varían los desempeños en el ámbito de programación, por una parte, y en el plano de las concepciones sobre la escritura, por otra, en función del año de ingreso. En cuanto a la primera de estas variables, la relación podría explicarse tanto por factores internos como externos. Estudios como el de Pérez, Hernández y González (2005) o García-Borrego y Córdoba-Cabús (2021) evidencian no solo la importancia de la motivación en el rendimiento académico, sino también cómo esta tiende a verse mermada con el paso de los cursos académicos, lo que podría explicar el descenso en su desempeño. En cuanto a la escritura, el hecho de que los mejores resultados se hallen en las primeras cohortes (2016 y 2017) puede tener relación con la madurez que en este ámbito se adquiere durante su proceso de formación profesional (VALDÉS-LEÓN, 2022).

En relación con los factores externos que podrían explicar el descenso en el rendimiento en función del año de ingreso, la bibliografía señala que, en muchas ocasiones, los alumnos de cursos terminales suelen asumir compromisos que les demandan mayor tiempo (cargas laborales o familiares), por lo que dedican menos horas semanales a sus estudios (ALCOVER y col., 2007; XAVIER y MENESES, 2022). Sumado a esto, es probable que los cambios en el itinerario formativo de la carrera puedan también incidir en el desempeño, pues se incluyeron, eliminaron y modificaron distintas asignaturas entre los años 2016 y 2022, lo que, de hecho, explica que coincidan en este curso estudiantes de diferentes cohortes.

Además de lo anterior, se observa una mejoría en el rendimiento de la tercera evaluación y, por ende, se estrechan los resultados entre esta dimensión y las percepciones de la escritura durante esta instancia. Uno de los aspectos que podría explicar esto es el hecho de que la evaluación 3 es la que posee un mayor componente procesual, por lo que es lógico pensar que quienes tienen una mirada de la escritura que valora su dimensión epistémica y compleja tengan mayores habilidades al momento de enfrentar evaluaciones de proceso.

La relación entre el proceso de escritura de textos y la escritura de código tiene varias similitudes epistemológicas, pero no han sido estudiadas previamente de manera empírica. Esta vinculación permitiría intercambiar y adaptar didácticas pedagógicas para contribuir con la mejora de la enseñanza de la producción tanto de textos como de códigos limpios. En este sentido, hemos analizado la relación empírica entre la percepción del proceso de escritura y las competencias de producción de software en tres etapas evaluativas. En la Figura 7, observamos la capacidad que tiene un modelo de regresión lineal de predecir la competencia de los estudiantes vinculada con establecer una metodología de resolución de problemas asociada a un producto de software. Si bien el modelo construido a partir de la percepción de la escritura ha sido capaz de predecir esta competencia de desarrollo, la relación no es significativa. Este resultado puede estar asociado a la estrategia de evaluación, la cual no está orientada a la producción ni de texto ni de código, sino de abstracción y conceptualización del problema.

De igual manera, en la Figura 8 se presenta un modelo construido con las mismas características de percepción del proceso de escritura, pero con el propósito de predecir la competencia de producir código limpio de una manera ordenada y utilizando estrategias técnicas basadas en patrones. La relación entre ambas dimensiones analizadas es más significativa que la anterior, considerando que, en esta oportunidad, los estudiantes debían producir (escribir) código a partir de un diseño previamente entregado. Luego, en la Figura 9 se presenta un modelo construido para predecir la competencia de producir código a partir de arquitecturas limpias, donde los estudiantes debían estimar el esfuerzo de escritura de código a partir de una arquitectura de software y, luego, escribir una porción de código. Es decir, los estudiantes debían realizar un proceso completo de producción, desde el diseño abstracto (arquitectura) hasta la codificación de una porción de dicho diseño. En esta oportunidad, la relación observada es aún más significativa que los modelos previos, alcanzado una alta capacidad de predicción del modelo. Ahora bien, considerando que esta investigación aborda el campo poco estudiado de la enseñanza de la escritura de código limpio y su relación con el proceso de escritura, resulta necesario reflexionar en torno a las similitudes que existen entre ambos procesos, pues nos parece que poner en diálogo estas disciplinas y su enseñanza puede resultar provechoso para la didáctica de cada una de ellas.

La didáctica de la escritura ha relevado el carácter procesual que esta posee desde hace ya varias décadas (DAVOODIFARD, 2022). Independientemente del foco que puedan tener los distintos modelos que existen para explicar y enseñar habilidades de escritura, ya sea desde perspectivas cognitivas (FLOWER y HAYES, 1996) hasta otras de carácter discursivo (BRONCKART, 2004), existe un consenso en que esta debe ser abordada como un proceso y no solo como un producto.

Así, es innegable que existe una larga tradición en cuanto a los estudios sobre la escritura, estudios que cobran mayor importancia desde la década de los 80' con los trabajos de Hayes y Flower (1980), Nystrand y col. (1982), De Beaugrande (1984), Bereiter y Scarmalia (1992), por nombrar algunos de los más influyentes. Este acervo ha dotado a los estudiosos de la enseñanza de la escritura de un sinnúmero de modelos teóricos y herramientas didácticas que han resultado fundamentales al momento de acompañar procesos de lectoescritura en el aula.

En contraste con esta larga tradición, la enseñanza de la programación ha cobrado especial relevancia en los últimos años, sobre todo debido al interés creciente de incorporar esta disciplina en la formación primaria y secundaria (DAPOZO y col., 2016; TORRENT MALUJE, 2019). Curiosamente, algunas de las experiencias didácticas relacionadas con la enseñanza de la programación suelen presentar aspectos muy cercanos a la didáctica de la escritura, tanto en el metalenguaje utilizado como en la perspectiva procesual de su enseñanza. Un ejemplo de ello es el extensivo uso de la metodología ADRI (Approach, Deployment, Result, and Improvement), la que se define como "an analytical tool which

is a well-known quality assurance model for self-review and external review and is used extensively in the education and business sectors" (MALIK y COLDWELL-NEILSON, 2017, p.1096). Las etapas de esta metodología se presentan a continuación en la Tabla 2, a la vez que se realiza un paralelo entre estas y la propuesta del grupo Didactext (DIDACTEXT, 2003).

Tabla 2. Comparación modelo ADRI para la enseñanza de la programación con el modelo Didactext para la enseñanza de la escritura.

ADRI	Didactext
(1) Approach - Thinking and planning	Acceso al conocimiento y Planificación
(2) Deployment - Implementing and doing	Producción textual
(3) Results - Monitoring and evaluating	
(4) Improvement - Learning and adapting	Revisión

Fuente: Elaboración propia.

La Tabla 2 representa una evidencia más de las similitudes que existen entre la enseñanza de una y otra disciplina. Dichas semejanzas van más allá de estos modelos didácticos particulares, pues representan un indicio de la cercanía que existe entre la producción de un código y la producción textual: algunos tecnicismos comunes como sintaxis, estructura, semántica y lenguaje dan cuenta de este vínculo. Pese a la relación entre ambas disciplinas y al potencial que esta posee en la enseñanza tanto de la programación como de la escritura, es muy poco lo que se ha publicado sobre ello. Algunos trabajos que abordan dicha cuestión son los de Santos (2020) y Lindgren (2021) quienes han esbozado el vínculo entre gramática y texto, y entre la escritura de código y la situación comunicativa, respectivamente. Sin embargo, como resultado de nuestra revisión bibliográfica, no ha sido posible encontrar trabajos que den cuenta de la estrecha relación entre escritura y programación que consideren no solo los aspectos más evidentes (como el metalenguaje común, por ejemplo), sino que indaguen en los procesos sociocognitivos involucrados con el fin de fortalecer su enseñanza.

En consecuencia, la producción de software es un proceso complejo que involucra el análisis del problema a resolver, el diseño de una arquitectura y la producción de código, idealmente, limpio (SOMMERVILLE, 2020). Para ello, es relevante la utilización de patrones o soluciones abstractas recurrentes en la producción de software, las que se encuentran tanto a nivel de diseño como de codificación (WEDYAN y ABUFAKHER, 2020). En este trabajo hemos observado cómo este proceso, en cada una de sus etapas, presenta una relación positiva con la percepción del proceso de escritura por parte de los estudiantes. En particular, cuando el proceso de producción de software se lleva a cabo de manera completa, la relación entre estas dos dimensiones de análisis se robustece (Figura 8).

6 Conclusiones

El objetivo de este trabajo fue relacionar la producción de código limpio con las concepciones sobre el proceso de escritura en estudiantes de Ingeniería informática. De esta forma, gracias a las tres evaluaciones prácticas que dieron seguimiento al proceso de producción de código limpio y al cuestionario sobre percepciones acerca de la escritura, se logró identificar tres aspectos clave: 1) la incidencia del año de ingreso en el desempeño académico, 2) la relación positiva entre producción de código limpio y percepciones sobre la escritura y las que se orientan hacia el desarrollo de competencias escritas y 3) las similitudes entre las propuestas didácticas para la enseñanza de la programación.

Respecto del primer hallazgo, a saber, la relación entre rendimiento y año de ingreso, la revisión bibliográfica señala que tanto factores personales como académicos pueden haber incidido en esto, aunque resulta necesario seguir indagando con el fin de dar respuesta ante esta necesidad formativa. Luego, en cuanto a la relación entre la producción de código limpio y las percepciones sobre el proceso de escritura, hemos identificado que existe una correlación significativa entre ambas variables, lo que podría explicarse debido a las similitudes que existen entre ambos procesos, tanto a nivel cognitivo como en el ámbito de su enseñanza.

Si bien el diseño y las características del estudio no permiten generalizar los resultados, estos invitan a profundizar en una interesante línea de investigación que se oriente hacia la didáctica,

que permita nutrir los procesos de enseñanza-aprendizaje de ambas disciplinas y que considere los aspectos epistémicos, sociocognitivos, procesuales y socioafectivos presentes tanto en el aprendizaje de la programación como en el de la escritura.

Referencias

ALCOVER, R y col. Análisis del rendimiento académico en los estudios de informática de la Universidad Politécnica de Valencia aplicando técnicas de minería de datos. In: XIII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2007). Teruel, España.: [s.n.], 2007. p. 163-170. Disponible en: http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2007/alanal.pdf.

ANICHE, Maurício y col. Code smells for model-view-controller architectures. *Empirical Software Engineering*, Springer, v. 23, n. 4, p. 2121-2157, 2018. DOI: 10.1007/s10664-017-9540-2. Disponible en: https://link.springer.com/article/10.1007/s10664-017-9540-2.

BASTÍAS, Oscar Ancán; DÍAZ, Jaime y RODRÍGUEZ, Cristian Olivares. Evaluation of Critical Thinking in Online Software Engineering Teaching: A Systematic Mapping Study. *IEEE Access*, IEEE, p. 167015-167026, 2021. DOI: 10.1109/ACCESS.2021.3135245. Disponible en: https://ieeexplore.ieee.org/abstract/document/9648189.

BECKER, Brett A y QUILLE, Keith. 50 years of cs1 at sigcse: A review of the evolution of introductory programming education research. In: PROCEEDINGS of the 50th acm technical symposium on computer science education. [S.l.: s.n.], 2019. p. 338-344. DOI: 10.1145/3287324.3287432.

BEREITER, Carl y SCARMALIA, Marlene. Dos modelos explicativos de los procesos de composición escrita. *Journal for the Study of Education and Development, Infancia y Aprendizaje*, Fundación Infancia y Aprendizaje, n. 58, p. 43-64, 1992. DOI: 10.1080/02103702.1992.10822332. Disponible en: https://www.tandfonline.com/doi/abs/10.1080/02103702.1992.10822332.

BRONCKART, Jean-Paul. *Actividad verbal, textos y discursos:: por un interaccionismo socio-discursivo*. [S.l.]: Fund. Infancia y Aprendizaje, 2004.

BROWN, WJ; MALVEAU, RC y col. *Architectures. anti-patterns: refactoring software and projects in crisis*. [S.l.]: John Wiley y Sons, 1998.

CALLE-ARANGO, Lina y ÁVILA REYES, Natalia. Obstacles, facilitators, and needs in doctoral writing: A systematic review. *Studies in Continuing Education*, Taylor & Francis, p. 1-19, 2022. DOI: 10.1080/0158037X.2022.2026315. Disponible en: https://www.tandfonline.com/doi/abs/10.1080/0158037X.2022.2026315.

DAPOZO, Gladys y col. Capacitación en programación para incorporar el pensamiento computacional en las escuelas. *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología*, SciELO Argentina, n. 18, p. 113-121, 2016. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=5889939.

DAVOODIFARD, Mahshad. An Overview of Writing Process Research: Using Innovative Tasks and Techniques for a Better Understanding of L2 Writing Processes in Assessment Contexts. *Studies in Applied Linguistics and TESOL*, v. 21, n. 2, 2022. DOI: 10.52214/salt.v21i2.8759. Disponible en: https://journals.library.columbia.edu/index.php/SALT/article/view/8759.

DE BEAUGRANDE, Robert. *Text production: Toward a science of composition*. [S.l.]: Praeger, 1984. v. 11.

DIDACTEXT, GRUPO DIDACTEXT Didáctica del Texto GRUPO. Modelo sociocognitivo, pragmalingüístico y didáctico para la producción de textos escritos. *Didáctica. Lengua y Literatura*, v. 15, p. 077-104, 2003. Disponible en: https://revistas.ucm.es/index.php/DIDA/article/download/DIDA0303110077A/19407.

DIETZ, Linus W y col. Teaching clean code. In: PROCEEDINGS of the 1st Workshop on Innovative Software Engineering Education. [S.l.: s.n.], 2018. Disponible en: http://ceur-ws.org/Vol-2066/isee2018paper06.pdf.

DIFABIO DE ANGLAT, Hilda. Hacia un inventario de escritura académica en el posgrado. *Revista de orientación educacional*, Universidad de Playa Ancha, n. 49, p. 37-53, 2012. Disponible en: https://dialnet.unirioja.es/descarga/articulo/4554495.pdf.

EGUILUZ, Andoni y col. Exploring the progression of early programmers in a set of computational thinking challenges via clickstream analysis. *IEEE Transactions on Emerging Topics in Computing*, IEEE, v. 8, n. 1, p. 256-261, 2017. DOI: 10.1109/TETC.2017.2768550. Disponible en: https://ieeexplore.ieee.org/abstract/document/8093668.

FIGUEIREDO, José y GARCÍA-PEÑALVO, Francisco José. Strategies to increase success in learning programming. In: IEEE. 2022 International Symposium on Computers in Education (SIIE). [S.l.: s.n.], 2022. p. 1-6. DOI: 10.1109/SIIE56031.2022.9982358. Disponible en: https://ieeexplore.ieee.org/abstract/document/9982358.

FINCHER, Sally. What are we doing when we teach programming? In: IEEE. FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings (IEEE Cat. No. 99CH37011). [S.l.: s.n.], 1999. v. 1, 12a4-1-12a4-5. DOI: 10.1109/FIE.1999.839268. Disponible en: https://ieeexplore.ieee.org/abstract/document/839268.

FLOWER, Linda y HAYES, J. *Textos en contexto*. Buenos Aires, Argentina: Asociación Internacional de Lectura, 1996. Disponible en: http://www.a43d.com.uy/jenny/wp-content/uploads/2018/06/Flowers_y_Hayes.pdf.

FOWLER, Martin. *Refactoring: improving the design of existing code*. [S.l.]: Addison-Wesley Professional, 2018.

GARCÍA-BORREGO, Manuel y CÓRDOBA-CABÚS, Alba. La transformación en la motivación y las aspiraciones de los estudiantes de periodismo a lo largo de tres años de grado: un análisis de las variables explicativas. *Vivat Academia. Revista de Comunicación*, n. 154, p. 23-41, 2021. DOI: 10.15178/va.2021.154.e1232. Disponible en: https://www.vivatacademia.net/index.php/vivat/article/view/1232.

GOLDSMITH, Rosalie; WILLEY, Keith y BOUD, David. Investigating invisible writing practices in the engineering curriculum using practice architectures. *European Journal of Engineering Education*, Taylor & Francis, v. 44, n. 1-2, p. 71-84, 2019. DOI: 10.1080/03043797.2017.1405241. Disponible en: https://www.tandfonline.com/doi/abs/10.1080/03043797.2017.1405241.

GONZÁLEZ, Martha Leticia Gaeta; GONZÁLEZ, Mercedes Zanotto y GONZÁLEZ-OCAMPO, Gabriela. Concepciones de escritura académica en estudiantes de medicina. *IE Revista de Investigación Educativa de la REDIECH*, Red de Investigadores Educativos Chihuahua AC, v. 11, p. 1-17, 2020. DOI: 10.33010/ie_rie_rediech.v11i0.855. Disponible en: https://www.redalyc.org/journal/5216/521662150034/521662150034.pdf.

GRIFFIN, Jean M. Designing intentional bugs for learning. In: PROCEEDINGS of the 1st UK & Ireland Computing Education Research Conference. [S.l.: s.n.], 2019. p. 1-7.

GROENEVELD, Wouter y col. Are Undergraduate Creative Coders Clean Coders? A Correlation Study. In: PROCEEDINGS of the 53rd ACM Technical Symposium on Computer Science Education V. 1. [S.l.: s.n.], 2022. p. 314-320. DOI: 10.1145/3478431.3499345.

GUPTA, Viral; KAPUR, Parmod Kumar y KUMAR, Deepak. Modelling and measuring code smells in enterprise applications using TISM and two-way assessment. *International Journal of System Assurance Engineering and Management*, Springer, v. 7, n. 3, p. 332-340, 2016. DOI: 10.1007/s13198-016-0460-0. Disponible en: https://link.springer.com/article/10.1007/s13198-016-0460-0.

HAYES, John R. y FLOWER, Linda S. *Identifying the organization of writing processes*. [S.l.]: Lawrence Erlbaum Assosiates, 1980.

IQBAL MALIK, Sohail y col. A web-based model to enhance algorithmic thinking for novice programmers. *E-Learning and Digital Media*, SAGE Publications Sage UK: London, England, v. 18, n. 6, p. 616-633, 2021. DOI: 10.1177/204275302110269. Disponible en: https://journals.sagepub.com/doi/full/10.1177/20427530211026988.

KILNER, Kerry; COLLIE, Natalie y CLEMENT, Jennifer. Using innovative teaching practices to inspire critically engaged reading and writing in a neoliberal university environment. *Higher Education Research & Development*, Taylor & Francis, v. 38, n. 1, p. 110-123, 2019. DOI: 10.1080/07294360.2018.1537258. Disponible en: https://www.tandfonline.com/doi/abs/10.1080/07294360.2018.1537258.

KYFONIDIS, Charalampos; MOUMOUTZIS, Nektarios y CHRISTODOULAKIS, Stavros. Block-C: A block-based programming teaching tool to facilitate introductory C programming courses. In: IEEE. 2017 IEEE Global Engineering Education Conference (EDUCON). [S.l.: s.n.], 2017. p. 570-579. DOI: 10.1109/EDUCON.2017.7942903. Disponible en: https://ieeexplore.ieee.org/abstract/document/7942903.

LACERDA, Guilherme y col. Code smells and refactoring: A tertiary systematic review of challenges and observations. *Journal of Systems and Software*, Elsevier, v. 167, pág. 110610, 2020. DOI: 10.1016/j.jss.2020.110610. Disponible en: https://www.sciencedirect.com/science/article/pii/S0164121220300881/pdfft?casa%5C_token=KJpj0zf7uFgAAAAA:H6BXM-kDoEqT%5C_GqUagfhDHRKZnUx9uA0xewQqU50T7Vgwy9PX3ciIR5jHA5a%5C_05-noRQRTvufCk%5C&md5=f259cb697bd1c02c96ae39cd7036fb65%5C&pid=1-s2.0-S0164121220300881-main.pdf.

LINDGREN, Chris Aaron. Writing With Data: A Study of Coding on a Data-Journalism Team. *Written Communication*, SAGE Publications Sage CA: Los Angeles, CA, v. 38, n. 1, p. 114-162, 2021. DOI: 10.1177/0741088320968061. Disponible en: https://journals.sagepub.com/doi/10.1177/0741088320968061.

LIU, Hui; JIN, Jiahao y col. Deep learning based code smell detection. *IEEE transactions on Software Engineering*, IEEE, v. 47, n. 9, p. 1811-1837, 2019. DOI: 10.1109/TSE.2019.2936376. Disponible en: https://ieeexplore.ieee.org/abstract/document/8807230.

LIU, Xiaohua y READ, John. General skill needs and challenges in university academic reading: Voices from undergraduates and language teachers. *Journal of College Reading and Learning*, Taylor & Francis, v. 50, n. 2, p. 70-93, 2020. DOI: 10.1080/10790195.2020.1734885. Disponible en: https://www.tandfonline.com/doi/abs/10.1080/10790195.2020.1734885.

LONKA, Kirsti y col. How to measure PhD students' conceptions of academic writing–and are they related to well-being? *Journal of Writing Research*, v. 5, n. 3, 2014. DOI: 10.17239/jowr-2014.05.03.1. Disponible en: https://www.jowr.org/index.php/jowr/article/view/688.

MALIK, Sohail Iqbal y COLDWELL-NEILSON, Jo. A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, Springer, v. 22, n. 3, p. 1089-1120, 2017. DOI: 10.1007/s10639-016-9474-0. Disponible en: https://link.springer.com/article/10.1007/s10639-016-9474-0.

MIRAS, Mariana; SOLÉ, Isabel y CASTELLS, Nuria. Creencias sobre lectura y escritura, producción de síntesis escritas y resultados de aprendizaje. *Revista mexicana de investigación educativa*, Consejo Mexicano de Investigación Educativa AC, v. 18, n. 57, p. 437-459, 2013. Disponible en: https://www.scielo.org.mx/pdf/rmie/v18n57/v18n57a6.pdf.

MURPHY, Ellen; CRICK, Tom y DAVENPORT, James H. An analysis of introductory university programming courses in the UK. *University of Bath*, 2016. Disponible en: https://purehost.bath.ac.uk/ws/portalfiles/portal/147312704.

NAVARRO, Federico y col. Panorama histórico y contrastivo de los estudios sobre lectura y escritura en educación superior publicados en América Latina. *Revista signos*, SciELO Chile, v. 49, p. 78-99, 2016. DOI: 10.4067/S0718-09342016000400006. Disponible en: https://www.scielo.cl/pdf/signos/v49s1/art06.pdf.

NYSTRAND, Martin y col. Rhetoric's "audience" and linguistics'"speech community": Implications for understanding writing, reading, and text. *What writers know: The language, process, and structure of written discourse*, Academic Press New York, p. 1-28, 1982.

OORT, Bart van y col. The Prevalence of Code Smells in Machine Learning projects. In: IEEE. 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN). [S.l.: s.n.], 2021. p. 1-8. DOI: 10.1109/WAIN52551.2021.00011. Disponible en: https://ieeexplore.ieee.org/abstract/document/9474395/.

PÉREZ, D; HERNÁNDEZ, Carmen Requena y GONZÁLEZ, Marta Zubiaur. Evolución de motivaciones, actitudes y hábitos de los estudiantes de la Facultad de Ciencias de la Actividad Física y del Deporte de la Universidad de León. *European Journal of Human Movement*, Asociación Española de Ciencias del Deporte, n. 14, p. 65-79, 2005. Disponible en: https://dialnet.unirioja.es/descarga/articulo/2279075.pdf.

PROKIĆ, Simona y col. Clean Code and Design Educational Tool. In: IEEE. 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO). [S.l.: s.n.], 2021. p. 1601-1606. DOI: 10.23919/MIPRO52101.2021.9597196. Disponible en: https://ieeexplore.ieee.org/abstract/document/9597196.

ROALD, Gunhild Marie y col. Learning from contrasts: first-year students writing themselves into academic literacy. *Journal of Further and Higher Education*, Taylor & Francis, v. 45, n. 6, p. 758-770, 2021. DOI: 10.1080/0309877X.2020.1813264. Disponible en: https://www.tandfonline.com/doi/full/10.1080/0309877X.2020.1813264.

RODRÍGUEZ HERNÁNDEZ, Blanca Araceli y GARCÍA VALERO, Laura Beatriz. Escritura de textos académicos: dificultades experimentadas por escritores noveles y sugerencias de apoyo. *Revista de Investigación Educativa*, Instituto de Investigaciones en Educación de la Universidad Veracruzana, v. 20, 2015. DOI: 10.25009/cpue.v0i20.1332. Disponible en: https://cpue.uv.mx/index.php/cpue/article/view/1332/2450.

SANTOS, André L. Javardise: a structured code editor for programming pedagogy in Java. In: CONFERENCE Companion of the 4th International Conference on Art, Science, and Engineering of Programming. [S.l.: s.n.], 2020. p. 120-125. DOI: 10.1145/3397537.3397561. Disponible en: https://dl.acm.org/doi/abs/10.1145/3397537.3397561.

SELWYN, Rebecca y RENAUD-ASSEMAT, Irene. Developing technical report writing skills in first and second year engineering students: a case study using self-reflection. *Higher Education Pedagogies*, Taylor & Francis, v. 5, n. 1, p. 19-29, 2020. DOI: 10.1080/23752696.2019.1710550. Disponible en: https://www.tandfonline.com/doi/full/10.1080/23752696.2019.1710550.

SOMMERVILLE, Ian. *Engineering software products*. [S.l.]: Pearson London, 2020.

TORRENT MALUJE, Catalina Andrea. *Programación computacional en escuelas: una mirada de inclusión y género a una iniciativa extracurricular del Ministerio de Educación de Chile*. [S.l.]: Universidad de Chile, 2019. Disponible en: https://repositorio.uchile.cl/handle/2250/170511.

TURKBEN, Tuncay. The Relationship between Fifth Grade Student's Writing Anxiety and Blocking with Their Written Expression Skills. *International Online Journal of Education and Teaching*, ERIC, v. 8, n. 2, p. 998-1021, 2021. Disponible en: https://hdl.handle.net/20.500.12451/8378.

ULU, Hacer. Investigation of Fourth Grade Primary School Students' Creative Writing and Story Elements in Narrative Text Writing Skills. *International Journal of Progressive Education*, ERIC, v. 15, n. 5, p. 273-287, 2019. DOI: 10.29329/ijpe.2019.212.18. Disponible en: https://ijpe.inased.org/makale/1126.

VALDÉS-LEÓN, Gabriel. Análisis de errores y variables sociolingüísticas: cómo escriben los estudiantes de primer año de universidad. *Onomázein*, Taylor & Francis, v. 58, 2022. DOI: 10.7764/onomazein.58.12. Disponible en: http://revistacienciapolitica.uc.cl/index.php/onom/article/view/58475.

VINE-JARA, Ana E. La escritura académica: percepciones de estudiantes de Ciencias Humanas y Ciencias de la Ingeniería de una universidad chilena. *Íkala, Revista de Lenguaje y Cultura*, Escuela de Idiomas, Universidad de Antioquia, v. 25, n. 2, p. 475-491, 2020. Disponible en: https://www.redalyc.org/journal/2550/255066610011/255066610011.pdf.

VOOGT, Joke y col. Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, Springer, v. 20, p. 715-728, 2015. DOI: 10.1007/s10639-015-9412-6. Disponible en: https://link.springer.com/article/10.1007/s10639-015-9412-6.

WALTON, Gabriel. Writing skills development in an engineering geology course through practice and feedback on report submissions using a rubric. *Journal of Geoscience Education*, Taylor & Francis, v. 68, n. 1, p. 33-48, 2020. DOI: 10.1080/10899995.2019.1625997. Disponible en: https://www.tandfonline.com/doi/abs/10.1080/10899995.2019.1625997.

WEDYAN, Fadi y ABUFAKHER, Somia. Impact of design patterns on software quality: a systematic literature review. *IET Software*, Wiley Online Library, v. 14, n. 1, p. 1-17, 2020. DOI: 10.1049/iet-sen.2018.5446. Disponible en: https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-sen.2018.5446.

WEINTROP, David. Block-based programming in computer science education. *Communications of the ACM*, ACM New York, NY, USA, v. 62, n. 8, p. 22-25, 2019. DOI: 10.1145/3341221.

WING, Jeannette M. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, The Royal Society London, v. 366, n. 1881, p. 3717-3725, 2008. DOI: 10.1098/rsta.2008.0118. Disponible en: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2008.0118.

XAVIER, Marlon y MENESES, Julio. Persistence and time challenges in an open online university: a case study of the experiences of first-year learners. *International Journal of Educational Technology in Higher Education*, SpringerOpen, v. 19, n. 1, p. 1-17, 2022. DOI: 10.1186/s41239-022-00338-6. Disponible en: https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-022-00338-6.

Contribuciones de los autores

Cristian Olivares-Rodríguez: Conceptualización, Curación de datos, Análisis formal, Software, Validación; **Gabriel Valdés-León**: Conceptualización, Análisis formal, Metodología, Validación; **Martha Vidal-Sepúlveda**: Conceptualización, Visualización, Redacción – borrador original, Redacción – Revisión y edición; **Romina Oyarzún-Yañez**: Conceptualización, Visualización, Redacción – borrador original, Redacción – Revisión y edición.