The teaching of programming: teachers' conceptions about introductory courses of computer science careers regarding the discipline

Enseñanza de la programación: concepciones de docentes de cursos introductorios de carreras de informática con respecto a la disciplina

Ana María Piccin[1] y Graciela Dora Susana Hadad[2]

[1] Universidad de Belgrano, CABA, Argentina
[2] Universidad Nacional del Oeste, Merlo, PBA, Argentina
ana.piccin@comunidad.ub.edu.ar,ghadad@uno.edu.ar

Abstract. The difficulty in learning programming is one of the notorious causes of abandonment in computer science careers. Despite various studies on the teaching and learning of programming, the specificities that make this difficulty are still not sufficiently clear. In this work, we study the perceptions of teachers of introductory programming courses in computer science careers at AMBA universities, regarding programming as a discipline. The use of the phenomenographic method allowed us to examine the conceptions that influence the decision-making of teachers, both in the planning of their courses and in the action in the classroom. A description of the conceptions shared by the teachers in the sample, regarding the discipline of programming, emerged from the systematic analysis of the teachers' expressions. Based on this collective conception of programming, the study made it possible to define some considerations towards educational management.

Keywords: Introductory teaching of programming, Teaching conceptions, Programming discipline, Phenomenography, Theory of didactics

Resumen. La dificultad en el aprendizaje de la programación es una de las causas notorias de abandono en carreras de informática. A pesar de diversos estudios sobre la enseñanza y el aprendizaje de la programación, no están aun suficientemente claras cuáles serían las especificidades que hacen a esa dificultad. En este trabajo se estudiaron las percepciones de los docentes de cursos introductorios de programación en carreras de informática de universidades del AMBA, respecto a la programación como disciplina. Para ello, la aplicación del método fenomenográfico permitió indagar sobre las concepciones que influyen en la toma de decisiones de los docentes, tanto en la planificación de sus cursos como en la

Received April 2022; Accepted Jun 2022; Published July 2022

acción en el aula. Se ofrece una descripción de la concepción con respecto a la disciplina de la programación compartida por los docentes que integraron la muestra, emergida del análisis sistemático de las expresiones de los docentes. En base a esta concepción colectiva de la programación, el estudio permitió definir algunas consideraciones hacia la gestión educativa.

Palabras Clave: Enseñanza introductoria de la programación, Concepciones docentes, Disciplina de la programación, Fenomenografía, Teoría de la didáctica

1 Introducción

La lectura de sucesivas versiones de The Joint Task Force [1] [2] [3] y de trabajos ofrecidos en congresos nacionales e internacionales [4] [5] [35], dan cuenta del interés por mejorar la enseñanza de la programación a nivel introductorio en las carreras de grado en informática. La relación entre fracasos en el aprendizaje y abandono temprano de las carreras de informática es una de las justificaciones mencionadas en los trabajos de investigación presentados en congresos [6] [7]. Los fracasos en los cursos de programación incidirían sustancialmente en el abandono temprano de las carreras de informática [7].

El objetivo principal de este trabajo es describir la concepción compartida, con respecto a la disciplina de la programación. Con este fin el estudio se ha enfocado en aquellos docentes que se desempeñan en universidades, tanto de gestión pública como privada, en el Área Metropolitana de Buenos Aires (AMBA). La pregunta que guía la investigación es: "¿Qué tipo de disciplina consideran que es la programación los docentes de los cursos introductorios en las carreras de grado de informática?"

El interés por estudiar las concepciones de los docentes con respecto a la disciplina de la programación surge de las siguientes observaciones:

- En el trabajo de Ahadi [7] se incluye una observación con respecto a las dificultades en la enseñanza de la programación que sugirió profundizar en el conocimiento de las especificidades de la disciplina: "(...) algo acerca de la programación la hace especialmente difícil de aprender a los principiantes y esto se refleja en la alta tasa de abandono en los niveles iniciales de los cursos de Ciencias de la Computación" [7].
- En Pears et al. [8] se observa que, si bien existen investigaciones con aportes interesantes para la enseñanza, éstos nunca llegarían a las aulas por la existencia de una brecha difícilmente salvable entre el campo de la educación y el de la informática. En el mismo sentido, Wilson y Guzdial [9] proponen: "…es necesario desarrollar investigaciones en el campo de la educación que permitan dar respuesta a preguntas que son propias del campo de la computación para favorecer el desarrollo de planes de mejora.".
- En Berglund et al. [10] se señala que, en general, las investigaciones surgidas desde las ciencias de la computación son de orden cuantitativo, con métodos de las ciencias naturales, mientras que las investigaciones surgidas desde las ciencias de la educa-

> ción son de orden cualitativo. Estos autores promueven la adopción de métodos cualitativos y mencionan especialmente la fenomenografía [8]. "Hay preguntas de investigación en educación informática relevantes donde una postura metodológica cualitativa permite que el investigador revele conocimientos que no son accesibles mediante abordajes generalmente utilizados en las ciencias naturales".

Un análisis de estas observaciones motivó el desarrollo de un estudio que permitiera describir cómo perciben la disciplina de la programación los docentes de los cursos introductorios de carreras de grado. Conocer la disciplina que se enseña es un requisito para el diseño de didácticas específicas [34]. El trabajo debería permitir obtener conclusiones de utilidad tanto para los docentes como para la gestión educativa.

La investigación se organizó sobre dos teorías, la de la educación y la de la informática. De educación se tomó la teoría de la enseñanza/aprendizaje y la de las concepciones de los docentes; de la informática, la teoría de la programación. La descripción de ambas teorías se presenta en la sección 2.

Para el diseño metodológico se apeló a la fenomenografía. Este método cualitativo fue diseñado especialmente para el tratamiento de concepciones de alumnos y docentes. En la sección 3 se describen las características de este método.

En la sección 4 se describe el desarrollo de la investigación. En la sección 5 se discuten los resultados del estudio en cuanto a la programación como disciplina según la conciben los docentes. Finalmente, se exponen conclusiones del trabajo en función de consideraciones hacia la gestión educativa en cuanto a la enseñanza y los contenidos de los cursos introductorios de programación, y se bosquejan futuros trabajos que amplíen el conocimiento respecto a la enseñanza introductoria de la programación.

2 Fundamento teórico

Para el diseño de la investigación se accedió a la teoría de las concepciones [12], a la teoría de la enseñanza y aprendizaje [13] y a la teoría de la programación [14] [15] [16], entre otros.

La didáctica de una disciplina debe dar cuenta necesariamente de los conocimientos propios de la misma. La didáctica de las matemáticas se ocupa de temas específicos como la demostración de teoremas o la intersección de conjuntos, la de historia de la relación entre las casas reinantes en Europa en el siglo XIX, la de lengua sobre la enseñanza de la estructura de la oración, por dar ejemplos.

Conocer la naturaleza de la disciplina de la programación y establecer un vínculo conceptual con otras disciplinas permitiría el estudio de didácticas, ya probadas, para determinar la factibilidad de su adaptación y su adopción en la enseñanza introductoria de la programación, para seleccionar técnicas y para transferirlas a los docentes.

2.1 Las concepciones didácticas

La actividad docente no es una actividad simplemente técnica, sino que requiere de cuotas de interpretación que, a su vez, exigen reflexión [13]. La enseñanza es referida como una actividad artística o por lo menos artesanal [13, p. 142]. Los procesos de

enseñanza y aprendizaje son procesos de interacción mental cuya riqueza reside precisamente en la singularidad subjetiva que los caracteriza. Las concepciones de los docentes influirán en sus decisiones con respecto a qué incluyen, qué excluyen, cómo transforman, qué estrategias pedagógicas son las adecuadas para enseñar la materia a sus alumnos. El valor de las transformaciones del saber, en lo que refiere al estudio de las concepciones didácticas, radica en que se trata de procesos inherentes a la práctica didáctica, ineludibles, que recorren desde la identificación del contenido a transmitir a los alumnos, hasta la verificación del aprendizaje.

Las concepciones didácticas se revelan durante la práctica de la enseñanza [12]. Cada docente ha tenido sus experiencias particulares, ha constituido sus propias concepciones, su propio saber enseñar [17]. El estudio de las concepciones didácticas implica un proceso de indagación y revelación de las impresiones, con respecto a la práctica de la enseñanza, de los docentes que participen en la investigación. La codificación de los relatos de los docentes, en términos de la teoría de la didáctica general y, en particular, de los modelos de enseñanza [13] [18], permitiría describir la manera en que los docentes enfocan la enseñanza introductoria.

2.2 La disciplina de la programación

Para fundamentar los aspectos de este estudio con respecto a la enseñanza de la programación se ha accedido a obras fundadoras de los autores Wirth [14], Dijkstra [15], Hoare [16], Naur [19] y Knuth [20].

La teoría de la programación, según Wirth [14], habría surgido cuando la atención de los teóricos se habría desplazado de la lógica de los algoritmos al estilo de su redacción. Enfocaron su estudio en la forma que adquieren los textos, los programas, en cuanto a expresión formal de los algoritmos. Especialmente de “aquellos textos de programas complejos que incluyen conjuntos de datos complicados” según Wirth [14, p. xii]. El programa es, y puede ser analizado, como un texto literario; tiene argumento y estilo.

La programación, analizada desde la práctica, es definida como una actividad de orden intelectual, de aplicación intensiva de lógica y abstracción que, si bien puede ser referida como “divertida” [21], como fácil para unos y como difícil para otros, es indudablemente compleja [14] [15] [16]. El contraste entre el atractivo de su práctica, en cuanto a la posibilidad que ofrece de producción de mundos abstractos sujetos a leyes diseñadas por el programador, y su fundamentación matemática, lógica y lingüística, sorprende al practicante novato. “Parece ser que la programación es una actividad intelectual con características únicas (…) el programador competente debe moverse entre varios niveles semánticos (…). [Se requiere] una agilidad desconcertante para quienes no están acostumbrados a ella” según [15, p. 611]. En los trabajos de Knuth [20] y Naggum [22], se otorga a la práctica de la programación el status de arte.

La eficiencia en la programación, así como su calidad, estarían directamente relacionadas con los conocimientos que el programador tiene de la implementación del lenguaje que está usando [16]. Cuanto más conozca de los conceptos sobre los que se

asienta el lenguaje de programación, más sencillo le resultará diseñar soluciones y programarlas. En Miller y Settle [23] se relaciona la dificultad para detectar y corregir errores con un conocimiento insuficiente del lenguaje, especialmente de la semántica.

A esto se debe agregar que, cuando se detectan errores durante la corrección de programas, el programador debe estar mentalmente dispuesto a desandar los caminos trazados durante el proceso constructivo para revisar la naturaleza de sus errores y producir nuevas abstracciones. A esto se refiere Dijkstra [15] como diferentes niveles de abstracción. Sobre esta característica fundamenta su afirmación de que la programación, en cuanto a práctica, es más compleja que la matemática.

El hecho de que la teoría de la programación se ocupara de programas complejos [14] [16], provocó el diseño de estándares para la práctica de la programación en entornos de producción. Se diseñaron las prácticas recomendadas, las que facilitarían la prueba y corrección de programas, las que permitirían probar programas complejos o repartir el trabajo de programación entre varios equipos de programadores. Las formas recomendadas de programación fueron acompañadas por el diseño e implementación de lenguajes de programación orientados a dichas prácticas.

Del análisis de estas obras podría bosquejarse la siguiente definición:

La programación de computadoras es una ciencia exacta [16], de naturaleza matemática [15], y en la que todas las propiedades de un programa y todas las consecuencias de ejecutarlo en cualquier entorno pueden, en principio, ser descubiertas a partir del texto del propio programa por medios de razonamiento puramente deductivo [14]; para su enseñanza podrían servir algunos modelos didácticos de la matemática y del lenguaje natural [19]; los procesos de enseñanza-aprendizaje de esta disciplina serían similares a los de una segunda lengua [24].

3 Metodología aplicada: fenomenografía

Para realizar el presente trabajo de investigación, se ha adoptado el método fenomenográfico [11] [8]. La fenomenografía es un método de investigación validado que comparte características metodológicas con la teoría fundamentada [26]. Fue diseñada por (Marton en) específicamente para el estudio de concepciones de estudiantes y docentes y se ha transformado en una herramienta idónea para la investigación en didáctica.

En la fenomenografía el foco del interés está puesto en las formas, y variedades de formas, en las que las personas experimentan y perciben los fenómenos, o aspectos de los fenómenos, que tienen lugar en los mundos en los que actúan. Los resultados son descriptivos y participan de un nivel colectivo, en el sentido de que los individuos son vistos como fragmentos de datos que contribuyen, entre todos, a constituir una experiencia colectiva e íntegra, que puede ser sujeta a estudio. El conocimiento se ve como la relación entre lo conocido y el cognoscente. La técnica de recolección de datos está basada en entrevistas. Es.as tienen como objeto lograr un entendimiento mutuo sobre el tema central del estudio. En general, se trata de entrevistas semiestructuradas.

Los datos se obtienen de una muestra de individuos tal que permita producir una variación exhaustiva de la experiencia. Dependiendo de la pregunta de investigación, los participantes son seleccionados de manera de obtener una muestra homogénea, o

bien extrayendo una muestra transversal que permita revelar la variación con que distintos individuos experimentan un fenómeno. El estudio fenomenográfico es crítico en el sentido que las entrevistas deben revelar un rango de perspectivas del fenómeno [25].

Los individuos son elegidos deliberadamente para cubrir la población de interés en dimensiones adecuadas para el estudio. La recolección de datos puede ser extendida si se considera que la variedad no ha sido efectivamente representada. También puede ser reducida, si se detecta que la incorporación de más material no es significativa [11].

Una vez finalizada la recolección de datos, se transcribe a texto las entrevistas realizadas, y los datos son extraídos del contexto individual de cada entrevista, es decir, se descontextualizan sus particularidades para lograr un "contexto colectivo de voces de otros individuos" [25, p. 17]. El investigador se involucra con el conjunto de los datos y busca diferencias críticas entre ellos que puedan actuar como catalizadores para la comprensión de la totalidad.

4 Desarrollo del trabajo

En esta sección se describe el proceso metodológico llevado a cabo para obtener la percepción colectiva de la disciplina. La Figura 1 muestra los módulos que conforman el estudio cualitativo.

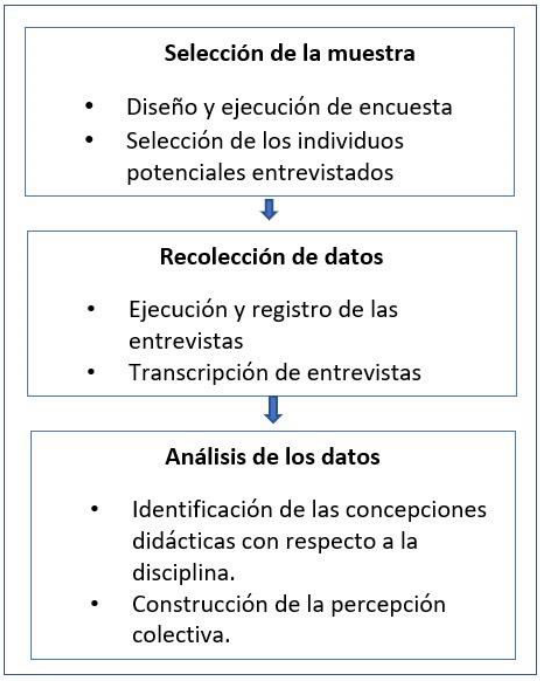

Fig. 1. Módulos del estudio cualitativo

La selección de la muestra incluyó una encuesta que permitió acceder a docentes de universidades del AMBA con carreras acreditadas por CONEAU. De entre los encuestados se seleccionaron los que fueron posteriormente entrevistados. La recolección de datos incluyó la ejecución y registro de las entrevistas con su correspondiente transcripción. El análisis de datos incluyó la identificación de las concepciones didácticas con respecto a la disciplina, que culminó con la construcción de la percepción colectiva de los docentes de los cursos introductorios de programación con respecto a la disciplina.

Este trabajo fue parte de una investigación con un objetivo de mayor alcance cuyos detalles están en [36].

4.1 La muestra

La muestra estuvo integrada por docentes de cursos introductorios de programación de universidades del AMBA.

Para evitar sesgos y asegurar su calidad los docentes fueron seleccionados de modo de abarcar la mayor variedad posible de experiencias en la enseñanza. Se tomó como criterio de inclusión a docentes que se desempeñaran en carreras de grado de informática, que estuvieran a cargo de cursos iniciales de programación, y cuyas carreras estuvieran acreditadas por la Comisión Nacional de Evaluación y Acreditación Universitaria. A este fin se diseñó una encuesta que fue respondida por 15 docentes de 10 universidades de gestión pública y privada con carreras de informática acreditadas, de un total de 29 universidades del AMBA.

El análisis de las respuestas en los cuestionarios resultó en la exclusión de la muestra de dos docentes, uno de ellos no se desempeñaba en una carrera específica de informática, y el otro no se desempeñaba en una carrera de grado. Sobre los individuos restantes se utilizó, como criterio de selección, que la población resultante representara la mayor variedad de perfiles. Finalmente, la composición de la muestra (ver Figura 2) fue la siguiente: el 20% tenía una experiencia en la enseñanza de la programación mayor a 20 años, el 30% tenía una experiencia de entre 15 y 20 años, el 10% tenía una experiencia de entre 10 y 15 años, el 30% tenía una experiencia superior a 5 años y el 10% una experiencia inferior a los 5 años.

El 90% de la muestra seleccionada desarrollaba actividad profesional, además de su desempeño académico, aplicando competencias de programación con una frecuencia relativamente alta. El 82% de la muestra recibió alguna formación pedagógica, ya sea en carreras de grado o realizando cursos. Se tuvieron 44 % de docentes actuando en universidades de gestión pública y 55 % en universidades privadas.

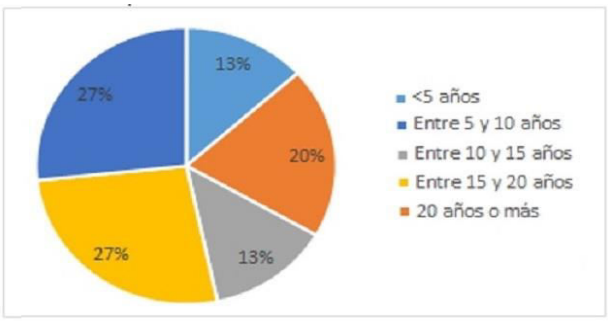

Fig. 2. Composición de la muestra

El método fenomenográfico no exige un número mínimo de individuos en la muestra; como en la teoría fundamentada [26], la muestra puede concluir por saturación. Se debe hacer primar la variedad sobre la cantidad, como mencionan Kinnunen y Malmi [27]: "(…) el foco de la investigación cualitativa no está en la amplitud de la muestra y la generalización de resultados. Un pequeño número de individuos es suficiente cuando los factores tienden a saturarse".

Este concepto fue aplicado para decidir la conclusión de la etapa de recolección de datos. Los docentes fueron entrevistados en un orden que permitiera garantizar la calidad de la muestra; se priorizó la variedad de sus perfiles. En paralelo con la ejecución de las entrevistas y su transcripción, se inició el proceso de análisis de datos. Esto permitió confirmar que se había alcanzado el punto de saturación de la muestra; es decir, nuevas entrevistas no aportaban nuevos datos.

4.2 La recolección de datos

Siguiendo el método fenomenográfico, se diseñó un protocolo organizado sobre preguntas semiestructuradas. El protocolo fue puesto a prueba en una prueba piloto, utilizando 3 docentes de la muestra inicial, haciéndole mínimos ajustes. La duración de la entrevista se estipuló en una hora, aceptando posibles desvíos impuestos por las circunstancias. A partir de la pregunta de investigación y del espacio conceptual del objeto de estudio, y siguiendo las recomendaciones metodológicas, se diseñó el siguiente cuestionario.

- Pregunta 1: ¿Cómo describiría la enseñanza de la programación? / ¿Qué es para usted enseñar la programar? / ¿Qué surge en su mente cuando piensa en enseñar las primeras herramientas de programación?
- Pregunta 2: ¿Qué es para usted la programación en cuanto a disciplina? / ¿Qué piensa de la programación en cuanto a disciplina? / ¿Qué tipo de disciplina es para usted la programación?
- Pregunta 3. ¿Qué espera que se lleven sus alumnos como aprendizaje de este curso? ¿Qué le gustaría que quedara en tus alumnos de este curso? ¿Cuándo se dice "está aprendiendo" o "ya aprendió"?

Las preguntas básicas se muestran acompañadas de preguntas auxiliares destinadas a facilitar la participación del entrevistado, sacarlo de puntos muertos o aclarar el sentido de lo que se pregunta [28] [29].

4.3 El análisis de datos

De acuerdo con el método fenomenográfico, el objeto bajo estudio, en este caso la disciplina debe emerger como una única voz que es una síntesis de todas las voces, que se expresaron en la muestra [11].

Cada entrevista fue transcripta textualmente. Las expresiones vertidas por los docentes fueron sometidas a codificación abierta y axial, bajo un proceso de comparación constante [26]. De la codificación abierta surgieron los ejes sobre los que se hizo el análisis de las expresiones de los entrevistados: concepciones acerca de la disciplina, de los resultados esperados, de la enseñanza y de los alumnos. La codificación axial permitió identificar las expresiones que reflejaban las concepciones sobre la disciplina a través de todos los ejes de análisis. Tales expresiones fueron volcadas en la primera columna de la Tabla 1. Mediante un proceso de comparación constante, a cada expresión se le asignó una categoría que se registró en la segunda columna de la Tabla 1.

Table 1. Concepciones de los docentes respecto a la disciplina de la programación.

Expresión del docente, relacionada con la disciplina	Categoría
1. Es una disciplina de orden práctico y hay que "sentarse en la computadora a hacer todos los ejercicios de la práctica".	• Disciplina de la práctica.
2. Se aprende con esfuerzo y práctica. 3. Se necesitan muchas "horas de máquina", de cuaderno, de lápiz, de libros, de hacer la práctica. "Programar no es soplar y hacer botellas"	• Requiere esfuerzo y constancia. • Disciplina de la práctica.
4. La disciplina de la programación exige abstracción de manera intensiva.	• Requiere ejercicio intensivo de la abstracción.
5. La complejidad de la programación es mayor que la del álgebra.	• Es más compleja que el álgebra.
6. Es una disciplina que "da apertura". Gracias a la programación muchas veces tienen apertura para otras cosas.	• Es una actividad que desarrolla el intelecto.
7. Algunos conceptos son determinantes en la posibilidad de desarrollar las competencias básicas de programación.	• Para aprender a programar se debe disponer de algunos conceptos que son determinantes.

Expresión del docente, relacionada con la disciplina	Categoría
8. "Se trata de organizar complejidades". 9. Para acercar la programación a las posibilidades de los alumnos es importante reducir la complejidad propia de esta disciplina.	• Disciplina compleja.
10. El nivel de abstracción requerido para el aprendizaje es superior al de otras disciplinas.	• Requiere un nivel de abstracción superior al de otras disciplinas.
11. La programación trata de resolver problemas: "[la programación] para mí es una herramienta para resolver problemas y una forma distinta de pensar de cómo solucionar los problemas".	• Desarrolla formas distintas de pensamiento. • Resolver problemas
12. La programación es una herramienta para resolver problemas, forma distinta de pensar, compleja, lógica.	• Disciplina compleja. • Requiere formas distintas de pensar. • Requiere uso de lógica.
13. Es comparable con matemática. En principio parecería ser más fácil.	• Sería más sencilla que la matemática.
"Tiene características distintivas": 14. hace pensar; 15. exige contacto permanente con el mundo real; 16. permite el aprendizaje a través del error; 17. el tipo de problemas que se tratan.	• Tiene características propias que la distinguen de otras disciplinas. • Requiere práctica.
18. Los tipos de problemas que se tratan en programación podrían ser característicos de la disciplina.	• Los problemas a resolver se caracterizan por tener soluciones algorítmicas.
19. "[El aprendizaje de la programación] es el desarrollo de competencias prácticas fundadas en estos conocimientos: algoritmos y estructuras de datos."	• Disciplina de la práctica. • Se funda en conceptos teóricos. • Algoritmos y estructuras de datos como herramientas para construir soluciones a los problemas.

Expresión del docente, relacionada con la disciplina	Categoría
20. Las dificultades que ofrece el aprendizaje de la programación no serían atribuibles, en principio, a un problema de inteligencia. "Podría tener que ver con tipos de inteligencia".	• La posibilidad de acceder a la programación dependería del tipo de inteligencia del alumno.
21. Requiere trabajo y dedicación	• Requiere esfuerzo y constancia.
22. Si bien la resolución de problemas exige capacidad de abstracción, para programar es además necesario moverse entre distintos niveles de abstracción, lo que añade complejidad.	• Requiere el manejo de distintos niveles de abstracción. • Disciplina compleja.
23. Se trata de representar el mundo físico en "un mundo que no lo es". 24. Aprender a programar da la capacidad de generar cosas.	• Requiere niveles de abstracción para trasladar el mundo real a una computadora

5 La disciplina de la programación según es percibida por los docentes

Mediante una revisión constante siguiendo el camino inverso, desde las concepciones registradas en la Tabla 1 hasta el texto original de las entrevistas, permitió identificar aspectos de valor para la gestión educativa, que se muestran en la Tabla 2.

Table 2. Categorías de concepciones según expresiones de los docentes.

Componentes	Nro. de expresión en Tabla 1
Disciplina de orden práctico	1, 2, 3, 15, 16, 19, 21
Requiere de niveles de capacidad de abstracción superiores a las de otras disciplinas; como por ejemplo el álgebra	4, 10, 22, 23
Resolución de problemas; problemas característicos; soluciones de orden algorítmico	11, 12, 17, 18, 19
Disciplina compleja	5, 8, 9, 12, 14, 15, 22, 24
Condiciones cognitivas e intelectuales	7, 10
Condiciones metacognitivas	2, 20
Desarrolla el intelecto	6, 11, 14

Las concepciones recogidas de la Tabla 1 y categorizadas en la Tabla 2 permiten formular la siguiente descripción de la disciplina, basada en docentes de cursos introductorios de programación de las carreras de informática de AMBA:

"La programación es una disciplina de orden práctico (1, 2, 3, 15, 16, 19, 21) que requiere de niveles de capacidad de abstracción superiores a las de otras disciplinas (4,

10, 22, 23); compleja, incluso más que el álgebra (5, 8, 9, 12, 14, 15, 22, 24). Su enseñanza/aprendizaje incluye la resolución de problemas, principalmente de carácter algorítmico. (11, 12, 17, 18, 19). Para su aprendizaje se requiere de capacidades determinadas cognitivas e intelectuales (7, 10) y también de capacidades metacognitivas como esfuerzo y constancia (2, 20); durante el aprendizaje se desarrolla el intelecto (6, 11, 14). Al compararla con la matemática, algunas veces aparece como más compleja (5), otras como más sencilla (13)".

Contrariamente a lo que se podría suponer, y a pesar de considerar que el aprendizaje de la programación requiere de capacidad de abstracción, de abstracción sostenida, y de aplicarla y administrarla en distintos niveles [15] los docentes no han relacionado a la disciplina de la programación en forma sustancial con la matemática. Sí han hecho referencia a conceptos básicos de matemática asociados a los de lógica, dentro de la misma expresión, en el mismo nivel de análisis, con mayor acento en los conocimientos básicos de lógica. Asimismo, han expresado que el conocimiento de elementos de matemática y de lógica favorecería su aprendizaje, pero la vinculación de programación y matemática no se extendió más allá, a diferencia de lo expresado en la literatura [14] [15] [16].

Esta consideración sobre la disciplina, en cuanto a que no depende fuertemente de la matemática, podría llevar a la gestión educativa a incluir la enseñanza de esta disciplina en los primeros cursos de las carreras de informática.

Uno de los docentes entrevistados consideró que no habría una relación causal entre la adquisición de contenidos de la matemática y el aprendizaje de la programación. Según este docente, no serían contenidos específicos de las matemáticas los que favorecerían el aprendizaje de la programación sino la forma de abordar el análisis de problemas y el diseño de soluciones, lo que sería coherente con el enfoque que adoptan los matemáticos. No se trataría de conceptos, en cuanto a contenidos específicos, sino de competencias. Esta concepción de la disciplina lo llevó a presentar a sus alumnos estrategias de resolución de problemas usadas en matemáticas para encarar el diseño de soluciones en programación, como lo es el identificar si el problema a resolver es un problema conocido y, de ser así, cuál sería el método aplicado para su resolución.

Este es el único entrevistado que se refirió expresamente a la formación en resolución de problemas durante el curso introductorio. En los demás casos, atendiendo a la calidad abstracta de los contenidos propios de la disciplina, los docentes coincidieron en buscar despertar el interés de los alumnos diseñando enunciados de problemas que incluyan situaciones concretas, de la vida real, a veces presentando problemas típicos de la profesión.

La organización de los talleres en equipos es vista por algunos docentes como una manera adecuada de introducir el uso de las buenas prácticas.

En circunstancias de diseñar estrategias de enseñanza, las concepciones didácticas con respecto a la disciplina de la programación pueden ser organizadas sobre dos ejes que pueden pensarse como ortogonales: uno práctico y otro analítico.

Las estrategias organizadas sobre el eje práctico pondrían el énfasis didáctico sobre la producción de programas; en las competencias para la construcción, redacción y prueba cobraría valor el aprendizaje de un lenguaje de programación. Las estrategias

organizadas sobre el eje algorítmico pondrían el énfasis didáctico en el estudio y aprendizaje de algoritmos clásicos; en las competencias para el diseño, la identificación y la adaptación de algoritmos a la construcción de soluciones de problemas cobrarían valor las capacidades intelectuales de análisis y de abstracción.

La visión práctica desplaza la verificación de la lógica algorítmica hacia la ejecución del programa. La visión algorítmica admitiría, en un extremo, demostraciones analíticas sin necesidad de que la solución algorítmica sea efectivamente implementada.

Los docentes entrevistados comparten la concepción de que la programación es una disciplina compleja que requiere aplicación intensiva de la abstracción y en mayor grado que la matemática. Para abordar las dificultades originadas por la calidad abstracta de los contenidos propios de la disciplina, los docentes adoptan la estrategia de reducir la complejidad de los enunciados para adaptarlos a las capacidades de los alumnos.

Esta estrategia docente, que muestra el interés por adecuar las prácticas a las necesidades de los alumnos, podría ser considerada por la gestión educativa para tomar decisiones respecto al diseño curricular, por ejemplo, incluir cursos preparatorios o niveladores dado que se trata de una disciplina compleja y abstracta. También podrían diseñarse cursos de capacitación docente sobre enfoques para articular los conocimientos previos de los alumnos con los requisitos que impone la disciplina, cumpliendo con los objetivos de los programas analíticos.

Con respecto a la evaluación de los aprendizajes, independientemente de los enfoques adoptados para la enseñanza, los docentes se han expresado con respecto a la conveniencia de evaluar durante el proceso. En este sentido, los docentes encuentran desarticulada su concepción de la disciplina con los sistemas de evaluación según los estándares académicos tradicionales [32] [33].

Esta consideración sobre la evaluación podría también ser revisada por la gestión académica para ajustar modalidades de evaluación a las características específicas de la disciplina. Por otro lado, se deberían estructurar los trabajos prácticos de manera que pudieran favorecer el obtener los resultados esperados y que colaboraran en la evaluación del proceso de aprendizaje.

La disciplina de la programación podría haber sido relacionada con el aprendizaje de un idioma extranjero, pero esto tampoco surgió de las entrevistas. Se asoció la disciplina con “construir nuevas realidades”, pero no específicamente al aprendizaje de un idioma [24]. Ni hubo entrevistado que se refiriera al hecho de que las palabras clave de los lenguajes de programación, aun estando en inglés, constituyeran una dificultad significativa para el aprendizaje.

Un conocimiento más profundo de esta disciplina permitiría formular objetivos adecuados y estrategias fundamentadas para decidir el nivel curricular en el que se encuentre incluida la programación introductoria en el plan de la carrera. De esta manera, se alinearían la modalidad de enseñanza y de evaluación con dichos objetivos.

6 Conclusión

Independientemente de que las raíces de la teoría de la programación puedan ser rastreadas hasta la matemática, la lógica y la lingüística, los docentes de los cursos introductorios conciben a la programación como una disciplina "de la práctica", en concordancia con Schön [31], aunque con un grado de complejidad mayor que otras disciplinas.

El concebir a la disciplina como "de la práctica" podría determinar un lugar en la institución y en los discursos institucionales inferior a las demás disciplinas. Para desarrollar estrategias de gestión educativa para la mejora de la calidad de la enseñanza introductoria de la programación, se requiere, por un lado, que su condición de práctica sea legitimada por la gestión educativa; y, por otro, que ésta se encuentre reflejada en los intereses y discusiones de la gestión a la par de disciplinas como matemática, lógica y lingüística [31].

La concepción de la disciplina como eminentemente práctica condicionaría a organizar los cursos introductorios sobre el eje del desarrollo de trabajos prácticos, complementada por la evaluación durante el proceso, estrategia de planificación compartida por los docentes. A diferencia de lo mencionado en [30], se deduce que la estructura de los cursos no se debería a una asimilación conceptual de la programación a la matemática, o a concebir a la competencia de la programación como derivadas de las matemáticas, sino a que la estructura de taller favorecería el aprendizaje en la práctica.

En definitiva, legitimar institucionalmente a la programación como una disciplina de la práctica implica reconocer la necesidad de disponer de un corpus práctico ad hoc que permita la reflexión en la acción para superar las "zonas indeterminadas de la práctica" [31]. Del mismo modo, se debería reconocer la necesidad de que las competencias específicas de los cursos introductorios de programación sean expresadas explícitamente como objetivos pedagógicos en los programas analíticos, que el repertorio de competencias a desarrollar se encuentre reflejado en los trabajos prácticos y que se tengan en cuenta las formas adecuadas de evaluación de aprendizajes.

Este trabajo de investigación sobre la enseñanza introductoria de programación se espera replicar en un área más amplia de carácter nacional para producir resultados generalizables que permitan colaborar en la mejora tanto de las prácticas docentes como en la toma de decisiones en la gestión educativa. Por otro lado, en una extensión de este trabajo podrían estudiarse las diferencias y similitudes de las concepciones docentes por áreas geográficas o por variantes de contexto académico, lo que podría derivar en recomendaciones diferenciadas en la gestión educativa.

Referencias

1. The Joint Task Force for Computing Curricula: Computing Curricula 2005: The Overview Report. ACM, AIS, IEEE-CS (2005).
2. The Joint Task Force on Computing Curricula: Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. ACM, IEEE Computer Society (2013).

3. ACM, IEEE-CS: Computing Curricula 2020: Paradigms for Global Computing Education. ACM, IEEE Computer Society (2020).
4. ITiCSE '20: Innovation and Technology in Computer Science Education Trondheim Norway 15 Junio, 2020- 19 Junio, 2020 Association for Computing Machinery, NY, USA, ISBN: 978-1-4503-6874-2 (2020).
5. ICER '20: International Computing Education Research Conference Virtual Event New Zealand 1 Agosto, 2020- 5 Agosto, 2020, Association for Computing Machinery, ISBN: 978-1-4503-7092-9 (2020).
6. Gorson, J., O'Rourke, E.: Why do CS1 students think they're bad at programming? Investigating self-efficacy and self-assessment. En: Proceedings of ITICSE´20, ACM (2020).
7. Ahadi, A.: Early Identification of Novice Programmers' Challenges in Coding Using Machine Learning Techniques. En: Proceedings of the 2016 ACM Conference on International Computing Education Research, pp. 263-264 (2016).
8. Pears, A. et al.: A Survey of Literature on the Teaching of Introductory Programming. En: ITiCSE-WGR '07: Working group reports on ITiCSE on Innovation and technology in computer science education, SIGCSE Bulletin, 39(4) (2007).
9. Wilson, C., Guzdial, M.: How to Make Progress in Computing. Communications of the ACM, 53(5), 35-37 (2010).
10. Berglund, A., Lister, R.: Introductory Programming and the Didactic Triangle. En: Clear, T., Jamer, J. (eds.) Conferences in Research and Practice in Information Technology, vol. 103 (2010).
11. Marton, F., Pong, W.Y.: On the unit of description in phenomenography. Higher Education Research & Development, 26(4), 335-348 (2005).
12. Camilloni, A.R.W.: Los profesores y el saber didáctico. En: Camilloni, A. (comp.) El saber didáctico. Paidós, Buenos Aires (2008).
13. Gimeno Sacristán, J., Pérez Gómez, A. I.: Comprender y transformar la enseñanza. Ediciones Morata, Madrid (2002).
14. Wirth, N.: Algorithms + Data Structures = Programs. Prentice-Hall Series in Automatic Computation, Prentice-Hall, Englewood Cliffs, New Jersey (1976).
15. Dijkstra, E.: Programming as a discipline of mathematical nature. En: The American Mathematical Monthly, 81(6), 608-612 (1974).
16. Hoare, C.A.R.: An Axiomatic Basis for Computer Programming. Communications of the ACM 1(10), p. 576 (1969).
17. Chevallard, Y.: La transposición didáctica. Del saber sabio al saber enseñado. Aique Grupo Editor, Buenos Aires (2009).
18. Joyce, B. R., Weil, M., Calhoum, E.: Models of Teaching. 9th. edn. Pearson (2017).
19. Naur, P.: Programming Languages, Natural Languages. and Mathematics. Communications of the ACM, 18(12), 137-148 (1975).
20. Knuth, D.: The Art of Computer Programming. Addison Wesley Publishing Company, Massachusetts (1969).
21. Watt, D.: Programming Language Design Concepts. John Wiley & Sons, Chichester, England (2004).
22. Naggum, E.: Ideas and principles. Programming (1996).
23. Miller, C.S., Settle. A.: Some Trouble with Transparency: An Analysis of Student Errors with Object-oriented Python. En: Proceedings of the 2016 ACM Conference on International Computing Education Research, pp. 133-141 (2016).
24. Solomon, J.: Programming as a Second Language. Learning & Leading with Technology, 32(4), 34-39 (2004).

25. Rands, M.L., Gansemer-Topf, A.M.: Phenomenography: A Methodological Approach for Assessing Student Learning in Student Affairs. The journal of affaire Inquiry, 1(2), 1-22 (2016).
26. Glaser, B. and Strauss, A.: The Discovery of Grounded Theory. Weidenfeld & Nicholson, London (1967).
27. Kinnunen, P., Malmi, L.: Why Students Drop Out CS1 Course? En: ICER'06, Canterbury, United Kingdom, pp. 97-108 (2006).
28. Trigwell, K.: A phenomenographic interview on phenomenography. En: Bowden, J.A., Walsh, E. (eds.) Phenomenography, RMIT Publishing, Melbourne, pp. 62-82 (2000).
29. Patrick, K.: Exploring conceptions: Phenomenography and the object of study. En: Bowden, J.A., Walsh, E. (eds.) Phenomenography. RMIT Publishing, Melbourne, pp.117-136 (2000).
30. Merchán Rubiano, S. M., López-Cruz, O., Gómez Soto, E.: Teaching computer programming: Practices, difficulties and opportunities. En: IEEE Frontiers in Education Conference (FIE). El Paso, TX, pp. 1-9 (2015).
31. Schön, D.: La formación de profesionales reflexivos. Paidós, Barcelona (1992).
32. Camilloni, A.R.W., Celman, S., Litwin, E., Palou de Maté, M.C.: La evaluación de los aprendizajes en el debate didáctico contemporáneo. 1ra. ed., Paidós, Buenos Aires – Barcelona - México (1998).
33. Ríos Muñoz, D.: Sentido, criterios y utilidades de la evaluación del aprendizaje basado en problemas. Educ Med Super, 21(3), La Habana (2007).
34. Camilloni, A.R.W.: Justificación de la didáctica. En: Camilloni, A. (comp.) El saber didáctico. Paidós, Buenos Aires (2008).
35. Becker, B.A., Quille, K.: 50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research. En: Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), ACM, New York, EEUU, pp. 338-344 (2019).
36. Piccin, Ana M., Enseñanza de la programación: las concepciones didácticas de los docentes de los cursos introductorios. En XXIII Edición del Workshop de Investigadores en Ciencias de la Computación, Chilecito-La Rioja, RedUNCI - UNdeC. ISBN: 978-987-24611-3-3, URL: https://drive.google.com/file/d/154tPVFoKFGe0WGATAPLAv0vc81RaPInt/view (2021).