ProgrEval: Una propuesta metodológica para la evaluación de conocimientos de programación

Tomás Caballero[1], Agustín Fernández-Ortúzar[1], Gonzalo Pablo Fernández[3,1] y Christian Cossio-Mercado[1,2]

[1] Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales. Departamento de Computación. Buenos Aires, Argentina.
{tcaballero,afernandezortuzar,ccossio}@dc.uba.ar

[2] CONICET-Universidad de Buenos Aires. Instituto de Ciencias de la Computación (ICC). Buenos Aires, Argentina.

[3] Universidad Nacional de Quilmes. Departamento de Ciencia y Tecnología. Bernal, Buenos Aires, Argentina.
gonzalo.pablo.fernandez@unq.edu.ar

Resumen Evaluar los conocimientos de los estudiantes es un desafío central para cualquier docente, y las clases de programación no son la excepción. Diseñar una evaluación requiere tener en claro su objetivo y el tipo de evidencias de aprendizaje que se desea obtener, entre otros aspectos, y asegurar que se ajuste a los contenidos y a la modalidad de enseñanza utilizada. Al analizar las propuestas disponibles se evidencia que no existe una opción consolidada, que sirva como base para evaluar conocimientos fundamentales de la programación, más allá de algunas iniciativas puntuales. Estas suelen situarse en dos extremos: por un lado, evaluaciones centradas en la resolución de ejercicios concretos en un lenguaje específico; por otro, pruebas que se asemejan más a evaluaciones cognitivas generales que a una evaluación específica del conocimiento en programación. En general, estas pruebas son estandarizadas y no contemplan las características de los estudiantes ni la forma en que se enseñaron los contenidos. Este trabajo presenta el marco general y los principales lineamientos de ProgrEval, una propuesta metodológica para la evaluación de conocimientos de programación en el nivel secundario, que incluye el diseño de instrumentos válidos y confiables, su puesta a punto y su mejora continua.

Palabras claves: Instrumentos de evaluación; Evaluación para el Aprendizaje; Enseñanza de Programación; Metodología de Evaluación

ProgrEval: A methodological approach for the evaluation of knowledge in programming

Abstract. Assessing students knowledge is a central challenge for any educator, and programming courses are no exception. Designing an assessment requires a clear understanding of its objectives and the type of learning evidence to be gathered, among other aspects, while ensuring alignment with both the content and the teaching approach used. After analysis of available approaches it is notable that there is no consolidated methodological proposal that serves as a foundation for evaluating fundamental programming knowledge, beyond a few isolated initiatives. These typically fall at two extremes: on one hand, assessments focused on solving concrete exercises in a specific programming language; on the other, tests that resemble general cognitive assessments more than evaluations of programming knowledge. In general, these assessments are standardized and fail to consider students characteristics or the way in which the content was taught. This work presents the general framework and key components of ProgrEval, a methodological proposal for programming assessment in secondary schools, which includes the design of valid and reliable instruments, their refinement, and their continuous improvement.

Keywords: Assessment instruments; Assessment for learning; Programming teaching; Assessment methodology

1. Introducción

La incorporación de las Ciencias de la Computación, y en particular de la programación, en las currículas educativas alrededor del mundo se viene desarrollando con mayor interés durante las últimas dos décadas, a raíz del impacto de la tecnología computacional en nuestra vida cotidiana. Hoy en día existen una gran variedad de iniciativas a cargo de países e instituciones que llevan adelante líneas de acción para incorporar estos contenidos en el aula, que incluyen la definición de propuestas curriculares, la producción de materiales y herramientas didácticas, y la formación de recursos humanos, entre otros. Para poder evaluar y medir el grado de alcance de los aprendizajes es necesario contar con instrumentos válidos que permitan recolectar evidencias de estos aprendizajes. Sin embargo, a diferencia de otras disciplinas STEM, como la física (Hestenes et al., 1992), que cuentan con evaluaciones estandarizadas, en nuestra disciplina las investigaciones sobre evaluación no abundan y no existe un consenso acerca de cuáles son los conocimientos fundamentales de programación que se pueden evaluar de manera general. En este sentido, en el área de didáctica de la programación existen una variedad de enfoques y trabajos que proponen instrumentos para evaluar estos conocimientos, algunos de los cuales presentaremos en este artículo. No obstante, estas propuestas ponen el foco en las características de estos instrumentos y su uso en diferentes contextos, y no en el proceso de diseño para poder confeccionarlos. En este trabajo nos ocuparemos de este último punto. El desarrollo de *ProgrEval* contempla los avances hasta el momento en el estudio de la evaluación de los conceptos fundamentales de programación, identificando puntos de mejora y propuestas para abordarlos con ideas fundamentadas en teorías de la didáctica general. Las bases conceptuales y la propuesta son útiles en cualquier nivel, pero algunas de las prácticas concretas recomendadas serían más adecuadas a partir del nivel medio en adelante.

2. Trabajos relacionados

Durante estos últimos años han surgido diversas propuestas que contemplan un amplio abanico de conceptos y habilidades a ser evaluadas. A partir de los trabajos analizados, organizamos estas propuestas en dos categorías: aquellas orientadas a la evaluación de conceptos fundamentales de la programación y aquellas orientadas a habilidades del pensamiento computacional. Román González et al. (Román González, 2015) se propusieron evaluar la habilidad de formular y resolver problemas mediante el uso de conceptos fundamentales de la computación y las herramientas de los lenguajes de programación: secuenciación, repetición, alternativas condicionales, funciones y variables. Brennan y Resnick (Brennan y Resnick, 2012) decidieron incluir más conceptos como la ejecución de programas en simultáneo y eventos, además de incluir prácticas que denominan computacionales como el desarrollo iterativo incremental, el diseño de casos de test y su aplicación para depurar programas, la reutilización y adaptación de código, y la generación de abstracciones para organizar y modularizar el código. A su vez, estaban interesados en evaluar el impacto de estas prácticas computacionales en los estudiantes con el objetivo de recolectar evidencias acerca de la mirada que habían formado sobre el uso de la computación como herramienta expresiva, de colaboración y de cuestionamiento sobre las tecnologías con las que interactuamos cotidianamente. Estos conceptos y prácticas involucradas están incluidas en lo que generalmente se conoce como Pensamiento Computacional (PC), término acuñado por Papert (Papert, 1980), retomado y resignificado años más tarde por Wing (Wing, 2006), con el objetivo de identificar aquellas habilidades cognitivas utilizadas por los científicos de la computación a la hora de resolver problemas, afirmando que se tratan de habilidades fundamentales aplicables a otros aspectos de la vida no relacionados con la computación. La Sociedad Internacional por la Tecnología en la Educación (ISTE) en conjunto con la Asociación de Profesores de Ciencias de la Computación (CSTA) formularon una definición operacional de PC para el nivel educativo K-12 (CSTA y ISTE, 2011) en la que, además de los temas mencionados, se incluyeron algunos como la capacidad de representar datos a través de abstracciones como modelos y simulaciones, de automatizar soluciones mediante el pensamiento algorítmico, de identificar,

analizar e implementar soluciones con el objetivo de alcanzar el uso óptimo de los recursos disponibles, y de generalizar procesos de resolución de problemas para transferirlos y aplicarlos a una gran variedad de estos. Además, incorporaron una serie de predisposiciones y actitudes deseables por parte de los estudiantes: confianza y persistencia al lidiar con la complejidad de un problema, tolerancia a la ambigüedad, y la habilidad de comunicarse y trabajar con otros para alcanzar un objetivo o solución común. En esta misma línea, Lockwood et al. (Lockwood y Mooney, 2018) proponen la utilización de desafíos Bebras [1] para confeccionar una prueba que aborde la resolución de problemas de la vida cotidiana aplicando habilidades de PC. De esta manera buscaban obtener una prueba independiente de la herramienta utilizada para la enseñanza. Desde otro enfoque, más ligado a los conceptos fundamentales de la programación que al pensamiento computacional, Tew y Guzdial (Tew y Guzdial, 2010) utilizaron un método para crear instrumentos de evaluación de conceptos fundamentales de programación, orientado a primeros años de carreras informáticas, que emplearon para elaborar dos implementaciones (Tew y Guzdial, 2011) (Parker et al., 2016). Este método contempla la validez del instrumento, la independencia del lenguaje y su reutilización. Grover (Grover, 2017) propuso utilizar un conjunto de instrumentos de evaluación que, combinados y suministrados en diferentes instancias, permitieran abarcar más habilidades y medir el desempeño de los estudiantes de manera más equitativa. Posteriormente en esta línea, desarrollaron una evaluación sumativa para cursos introductorios de programación (Grover, 2020), orientada a estudiantes de 11 a 14 años de edad, siguiendo una metodología de diseño que contempla conceptos y habilidades fundamentales de programación. En resumen, cada una de las propuestas mencionadas ofrecen herramientas para pensar, diseñar e implementar una evaluación de conocimientos de programación. Luego de analizar estos trabajos, surgen algunos interrogantes que motivan la revisión, modificación y extensión de estas metodologías para poder diseñar una propuesta que contemple un conjunto de características cuyo desarrollo consideramos pertinente profundizar.

3. Críticas generales

Al indagar en el estado del arte del desarrollo de evaluaciones, identificamos una tendencia a producir nuevos instrumentos de evaluación que atiendan a diferentes necesidades y no al desarrollo de una metodología que facilite su confección. En particular, observamos que algunas características propias de un proceso de evaluación están siendo desatendidas y pretendemos que tengan un rol protagónico en *ProgrEval*. Por ejemplo, algunas de estas evaluaciones intentan evaluar habilidades generales y no reparan enlas trayectorias y conocimientos previos de los estudiantes (e.g., (Lockwood y Mooney, 2018), (Román González, 2015)), mientras que otras se limitan a evaluar aprendizajes sobre un entorno o lenguaje (Grover, 2020), dificultando la tarea de extraer evidencias sobre conceptos o habilidades que puedan ser comparables. En línea con esto, observamos que al emplear instrumentos para evaluar habilidades del pensamiento computacional surgen diferentes problemáticas debido a la falta de consenso sobre su definición (Brennan y Resnick, 2012) (Román González, 2015) (CSTA y ISTE, 2011) y de argumentos que relacionen los procesos de aprendizaje con las habilidades que se quieren evaluar (Dapozo et al., 2022). En relación a la validación de los instrumentos, observamos que en la mayoría de los casos sólo se verifica la validez de constructo de los instrumentos, ignorando otros aspectos relevantes de la validez, como la confiabilidad o la convergencia (de Camilloni et al., 2008). Además, se pone el foco en la practicidad y universalidad de los instrumentos, derivando principalmente en evaluaciones estandarizadas que, si bien su uso está muy propagado, han recibido muchas críticas en las últimas décadas por parte de la comunidad educativa respecto de su validez (Anijovich y Cappelletti, 2018) (Gómez Yepes, 2010) (Benítez y Gamboa, 2022) (Moreno Olivos, 2016) (Popham, 1999) . Señalan, entre otras cuestiones, cómo las evaluaciones estandarizadas no tienen en cuenta el contexto de los estudiantes, el modo en que se les enseñó y su progreso a lo largo del tiempo. Respecto a los instrumentos resultantes, la mayoría de las

[1] Bebras International Challenge on Informatics and Computational Thinking: https://www.bebras.org

propuestas optan por consignas de opción múltiple y ,en algunos casos, consignas de respuesta abierta de longitud limitada, con el objetivo de agilizar las correcciones y devoluciones. Sin embargo, es importante contar con herramientas de evaluación que se amolden a las necesidades de aprendizaje de los estudiantes además de las limitaciones prácticas de los docentes, que provean la evidencia necesaria dado un contexto y posibilidades específicas. En línea con esto, algunas propuestas han señalado la dificultad de las consignas y la cantidad de tiempo requerido para completar una evaluación en su totalidad como factores determinantes en los resultados obtenidos (Parker et al., 2021). En general, las propuestas analizadas ponen el foco en los instrumentos y no en lineamientos que permitan confeccionar un instrumento de evaluación acorde al contexto de aprendizaje, considerando las trayectorias de los estudiantes, que incluya un plan de mejora continua de los instrumentos y otorgue evidencias para comparar resultados.

4. Propuesta de trabajo

La propuesta *ProgrEval* tiene por objetivo facilitar el proceso de confección de un instrumento de evaluación de modo que contemple las trayectorias de los estudiantes e incorpore procesos de validación y mejora continua. En los trabajos antes analizados encontramos herramientas e insumos que nos parece pertinente tener en cuenta para alcanzar dichos objetivos. Partiendo de algunas de las propuestas principales disponibles (e.g., (Tew y Guzdial, 2010), (Grover, 2020), (Román González, 2015)), concentradas en el uso y mejora de los instrumentos elaborados, destacamos algunos lineamientos comunes que identificamos en estos procesos: la definición de un núcleo de conceptos fundamentales a evaluar, objetivos de aprendizaje asociados a estos conceptos, el diseño de consignas y la validez del instrumento. Decidimos organizar la propuesta en torno a la validez de **contenido**, de **construcción** y de **convergencia**, los tres tipos de validez que consideramos más relevantes dentro de los definidos por Camilloni (de Camilloni et al., 2008), aunque sin perder de vista a los demás. Para trabajar sobre la validez de **contenido**, entendida como el grado en que el contenido de una evaluación es una muestra significativa del contenido enseñado, definimos un núcleo de *conceptos fundamentales de la programación* y *objetivos de aprendizaje* asociados a estos. Algunos de los conceptos fundamentales que hemos considerado son: repetición, alternativa condicional, variables, funciones, procedimientos y estructuras de datos. El objetivo de esta parte es brindar a los docentes un marco conceptual que organice los contenidos que desean evaluar para poder tenerlos presentes durante el desarrollo de las consignas de evaluación. Luego, para trabajar sobre la validez de **construcción**, comprendida como el grado de correspondencia entre las estrategias de enseñanza utilizadas, las teorías que explican los procesos de aprendizaje de los alumnos, y la selección y diseño de los instrumentos de evaluación, establecemos un conjunto de *desempeños de programación* que comprenden aspectos del tratamiento de programas, como la producción o la interpretación de código, y de la resolución de problemas, como la identificación de patrones o división en subtareas. Con base en estos desempeños, seleccionamos *tipos de consignas* que los ponen en práctica (Grover, 2021) (Ruf et al., 2015) (Simões y Queirós, 2020). Estas consignas se presentan de manera independiente al lenguaje o entorno utilizados para la enseñanza, permitiendo diseñar la evaluación minimizando las limitaciones o complejidades accidentales que éstas puedan introducir. Por último, para trabajar sobre la validez de **convergencia**, que concierne a la relación que existe entre un instrumento de evaluación y otros instrumentos de validez ya conocidos, establecemos una serie de lineamientos para la adaptación de un instrumento ante cambios en el contexto, y para su mejora luego de su aplicación. Para esto último, presentamos algunas técnicas estadísticas como el análisis factorial para determinar la consistencia entre consignas, y los índices de discriminación para determinar qué tan bien una consigna puede distinguir entre los estudiantes que cumplieron los objetivos de aprendizaje de aquellos que no. Sin embargo, pretendemos aportar orientaciones que les permita a los docentes llevar a cabo un proceso de mejora menos riguroso pero efectivo. Para esto, nos interesa abordar la idea de una meta-evaluación, empleando parte de las ideas que se proponen en el marco de la evaluación para el aprendizaje, para identificar qué aprendizajes hay que mejorar y qué acciones se pueden

llevar a cabo para mejorar, en este caso, los instrumentos. En resumen, nuestra propuesta de trabajo considera los aportes y esfuerzos realizados durante las últimas décadas para la evaluación de conocimientos fundamentales de programación, y pone el foco en el desarrollo de una propuesta metodológica que considere el proceso de evaluación en su totalidad, desde la definición de un núcleo de conocimientos fundamentales hasta la mejora continua de los instrumentos.

Referencias

Anijovich, R., & Cappelletti, G. (2018). *La evaluación como oportunidad*. Buenos Aires, Argentina: Paidós.

Benítez, J., & Gamboa, L. A. (2022). Evaluación estandarizada de los aprendizajes: una revisión sistemática de la literatura. *CPU-e, Revista de Investigación Educativa*. https://doi.org/10.25009/cpue.v0i34.2800

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 annual meeting of the American Educational Research Association*, 1-25.

CSTA & ISTE. (2011). Operational Definition of Computational Thinking for K-12 Education. https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf

Dapozo, G. N., Greiner, C. L., Petris, R. H., Irrazábal, E., Company, A. M., Espíndola, M. C., & Medina, Y. (2022). Evaluación de habilidades de pensamiento computacional al inicio de una asignatura de programación en una carrera de informática, 137-146.

de Camilloni, A. R. W., Cehnan, S., Litwin, E., & Palou de Maté, M. d. C. (2008). *La Evaluación de los aprendizajes en el debate didáctico contemporáneo*. Paidós.

Gómez Yepes, R. L. (2010). Calidad Educativa: Más que resultados en pruebas estandarizadas.

Grover, S. (2017). Assessing Algorithmic and Computational Thinking in K-12: Lessons from a Middle School Classroom. En P. J. Rich & C. B. Hodges (Eds.), *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 269-288). Springer International Publishing. https://doi.org/10.1007/978-3-319-52691-1_17

Grover, S. (2020). Designing an Assessment for Introductory Programming Concepts in Middle School Computer Science. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 678-684. https://doi.org/10.1145/3328778.3366896

Grover, S. (2021). Toward A Framework for Formative Assessment of Conceptual Learning in K-12 Computer Science Classrooms. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 31-37. https://doi.org/10.1145/3408877.3432460

Hestenes, D., Wells, M., & Swackhamer, G. (1992). Force concept inventory. *The Physics Teacher*, *30*(3), 141-158. https://doi.org/10.1119/1.2343497

Lockwood, J., & Mooney, A. (2018). Developing a Computational Thinking Test using Bebras problems. *TACKLE: the 1st Systems of Assessments for Computational Thinking Learning workshop at EC-TEL 2018*.

Moreno Olivos, T. (2016). Las pruebas estandarizadas en la escuela contemporánea, £llave o cerrojo para la mejora de la educación? *Temas de Educación*, *22*, 83-96.

Papert, S. A. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA.

Parker, M. C., Guzdial, M., & Engleman, S. (2016). Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment. *Proceedings of the 2016 ACM Conference on International Computing Education Research*, 93-101. https://doi.org/10.1145/2960310.2960316

Parker, M. C., Guzdial, M., & Tew, A. E. (2021). Uses, Revisions, and the Future of Validated Assessments in Computing Education: A Case Study of the FCS1 and SCS1. *Proceedings of the 17th ACM Conference on International Computing Education Research*, 60-68. https://doi.org/10.1145/3446871.3469744

Popham, W. (1999). Why Standardized Test Scores Don't Measure Educational Quality. *Educational Leadership*, *56*, 8-15. https://api.semanticscholar.org/CorpusID:140978534

Román González, M. (2015). Computational Thinking Test: Design Guidelines and Content Validation. En *Proceedings of the 7th Annual International Conference on Education and New Learning Technologies (EDULEARN 2015)* (pp. 2436-2444). IATED, Barcelona, Spain. https://doi.org/10.13140/RG.2.1.4203.4329

Ruf, A., Berges, M., & Hubwieser, P. (2015). Classification of Programming Tasks According to Required Skills and Knowledge Representation. En V. J. Brodnik A (Ed.), *Informatics in Schools. Curricula, Competences, and Competitions* (pp. 57-68). Heidelberg: Springer.

Simões, A., & Queirós, R. (2020). On the Nature of Programming Exercises. *arXiv preprint*. https://doi.org/10.48550/arXiv.2006.14476

Tew, A. E., & Guzdial, M. (2010). Developing a validated assessment of fundamental CS1 concepts. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 97-101. https://doi.org/10.1145/1734263.1734297

Tew, A. E., & Guzdial, M. (2011). The FCS1: a language independent assessment of CS1 knowledge. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 111-116. https://doi.org/10.1145/1953163.1953200

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