12

Propuesta metodológica para incorporar la Inteligencia artificial generativa en la formación universitaria de programación

Methodological proposal to incorporate generative Artificial Intelligence in university programming training

Dr. Jesús Sánchez Allende
CUNEF Universidad, España

Resumen
Este trabajo presenta una propuesta sobre el uso de la IA generativa en la universidad dada la carencia de marcos metodológicos pedagógicamente fundamentados. Se propone una metodología de cuatro fases secuenciales que integra evidencia de meta estudios en tutoría inteligente, retroalimentación automatizada, personalización adaptativa y LLM en formación de programación. Se identifican cinco componentes clave cuya integración sistemática evidencia un impacto en el aprendizaje: retroalimentación inmediata contextualizada, personalización según el nivel de conocimientos, andamiaje cognitivo progresivo, multimodalidad en la presentación de contenidos y análisis predictivo. Las evidencias recopiladas, a partir principalmente de meta estudios anteriores, proporcionan las bases para definir la propuesta metodológica. En la propuesta también se abordan algunas de las limitaciones que podemos encontrar en la literatura como la dependencia excesiva de las herramientas de IA y la necesidad de formación del docente. La propuesta va dirigida al desarrollo de autonomía y pensamiento crítico de los estudiantes.

Palabras clave: IA generativa, programación, educación universitaria, retroalimentación, personalización adaptativa.

Suggested citation:
Sánchez Allende, J. (2025). Propuesta metodológica para incorporar la Inteligencia artificial generativa en la formación universitaria de programación. In Actis Di Pasquale, E. (Editor), *Artificial intelligence in education: applications, proposals and challenges*. (pp. 136-147). Madrid, España: Adaya Press. https://doi.org/10.58909/ad2560369

Abstract
This book chapter presents a proposal regarding to the use of generative AI in university education, addressing the current scarcity of pedagogically grounded methodological frameworks. A four-phase sequential methodology is proposed, integrating evidence from meta-studies on Intelligent Tutoring Systems (ITS), automated feedback, adaptive personalization, and Large Language Models (LLMs) within the context of programming training. Five key components are identified, the systematic integration of which demonstrates a tangible impact on learning: contextualized immediate feedback, personalization based on proficiency levels, progressive cognitive scaffolding, multimodality in content presentation, and predictive analytics. The gathered evidence, derived primarily from prior meta-analyses, provides the foundation for defining this methodological proposal. Furthermore, the proposal addresses limitations identified in the literature, such as the potential for over-reliance on AI tools and the imperative for faculty training. Ultimately, the proposed framework is aimed at fostering student autonomy and critical thinking.

Keywords: generative AI, programming, university education, feedback, adaptive personalization.

Introducción

La integración de modelos de inteligencia artificial en la educación universitaria ha experimentado un crecimiento exponencial desde la aparición de ChatGPT en noviembre de 2022, transformando la enseñanza-aprendizaje. En el ámbito específico de la programación, esta aparición plantea tanto oportunidades sin precedentes como desafíos metodológicos complejos que requieren un enfoque sistemático y riguroso para su implementación efectiva.

La literatura sobre tecnología educativa documentada en metaanálisis contemporáneos revela patrones consistentes. Zhang, Jantakoon, & Laoha (2025) en su revisión sistemática de sistemas de tutoría inteligente (ITS) muestran efectos positivos consistentes cuando estas herramientas se diseñan con principios pedagógicos explícitos. Tan, Rudolph, & Tan (2024) extiende estos hallazgos específicamente a contextos de programación, documentando que los efectos más grandes se logran cuando la tecnología se integra con supervisión docente estratégica, no cuando reemplaza al docente.

La educación en programación presenta características únicas que diferencian sus desafíos de otras disciplinas STEM para dar soporte a las competencias técnicas, el pensamiento computacional y la resolución de problemas algorítmicos. Callejo Pinardo, Alario Hoyos, & Delgado Kloos (2024) en su análisis de dificultades conceptuales en programación documentan que los errores más frecuentes no son sintácticos sino conceptuales: incomprensión de paradigmas, confusión en la lógica de control, y falta de pensamiento algorítmico abstracto. Estos desafíos requieren intervenciones pedagógicas especializadas.

En programación, los estudios sistemáticos más actuales muestran resultados variables. Pitts, Hridi, & Narayanan (2025) identificaron tres áreas críticas sobre aplicaciones de sistemas LLM en programación: retroalimentación formativa de código, evaluación automatizada y modelado del conocimiento. Complementariamente, Reihanian, Hou, Chen, & Zheng (2025) destacaron la importancia del equilibrio entre automatización y supervisión humana, identificando como principales preocupaciones la precisión, autenticidad y evaluación. Yan, Chen, Hu, & Ye (2025) mostraron que la programación colaborativa asistida por LLM mejora significativamente las habilidades de pensamiento computacional y reduce la carga cognitiva de los estudiantes.

A pesar de disponer de evidencias positivas, falta un marco metodológico que sintetice estas evidencias. El objetivo es proporcionar un modelo que facilite la integración efectiva y pedagógicamente fundamentada de modelos de IA en el aprendizaje universitario de programación, considerando tanto las oportunidades como los riesgos inherentes identificados en la literatura.

Estudio y análisis de propuestas previas

En el estudio de las evidencias se han consultado las bases de datos: Scopus, Web of Science, ACM Digital Library y Google Scholar con criterios de publicaciones entre 2022-2025 para metaanálisis y estudios en el ámbito universitario centrado en la formación de programación. Se han excluido artículos de experiencias sin datos y los artículos sobre educación secundaria. Se han evidenciado cuatro grandes áreas: la tutoría inteligente, la retroalimentación automatizada, la personalización y el aprendizaje adaptativo y el uso de modelos grandes de lenguaje (LLM).

Tutoría inteligente (ITS) en educación de programación

Los Sistemas de tutoría inteligente (ITS) se han ido incorporando en el entorno docente. El exhaustivo metaanálisis realizado por Huang, Xu, & Liu (2025), donde revisan la evidencia más actual, identifican un tamaño del efecto global de (d = 0,86) magnitud considerada "grande". En este estudio se confirma que los ITS pueden mejorar significativamente el aprendizaje y la evaluación en pruebas estandarizadas, superando las limitaciones observadas en revisiones anteriores.

Sin embargo, al concretarse en competencias de alto nivel cognitivo, los resultados hay que considerarlos con cuidado en la enseñanza de la programación. Huang et al., (2025) indican que, aunque los beneficios sobre los conocimientos declarativos son evidentes, los efectos sobre la "adquisición de habilidades de resolución de problemas" y el "desarrollo práctico" son menos concluyentes. El estudio sugiere que la eficacia en estas áreas complejas depende del diseño instruccional indicando que los ITS que integran estrategias como ejemplos resueltos y elementos de gamificación consiguen un impacto positivo superior.

Retroalimentación automatizada y calificación automática

La retroalimentación es uno de los mecanismos de mayor impacto. Hattie & Timperley (2007) en su metaanálisis documentan que la retroalimentación de calidad supone un gran impacto, pero su efectividad depende críticamente de tres elementos: el objetivo, la situación del estudiante respecto al objetivo y cómo se reduce dicha brecha.

La eficacia de la retroalimentación no se debe a la presencialidad, sino a la calidad informativa y la temporalidad. En el metaanálisis de Wisniewski, Zierer, & Hattie (2020), se revisaron 435 estudios donde se concluye que el impacto en el aprendizaje varía según la especificidad: la retroalimentación basada en el simple refuerzo o corrección genérica tiene un efecto bajo, la que es altamente informativa y con andamiaje cognitivo apropiado llega a un efecto de (d = 0.99).

La calificación automatizada es cada vez más relevante. Paiva, Leal, & Figueira (2022), en su revisión del estado del arte, destacan que estas herramientas permiten una evaluación formativa continua, reduciendo de forma significativa el tiempo de corrección y ayudando al estudiante con retroalimentación inmediata lo que no es factible con la formación presencial.

Personalización y aprendizaje adaptativo

La teoría de carga cognitiva (Sweller, 2020) establece que la personalización reduce la carga cognitiva al presentar información en el nivel correcto de complejidad para cada estudiante específico. Mayer (2024) en sus principios de aprendizaje multimedia documenta que múltiples representaciones de un concepto (texto, diagrama, video, código ejecutable) permiten reducir la carga cognitiva al compararlo con una sola modalidad.

En St-Hilaire et al. (2022) se presenta que una plataforma como *AI Korbit* consigue que los estudiantes obtengan puntuaciones hasta 2,5 veces superiores en comparación con cursos en MOOC no adaptativos. Este tipo de experiencias muestra el potencial del aprendizaje adaptativo sobre métricas como el tiempo dedicado por los estudiantes, la tasa de finalización, la retención de conocimientos y la precisión, mejorando la experiencia global de aprendizaje.

Modelos grandes de lenguaje (LLM)

ChatGPT y modelos similares representan un cambio significativo en el ámbito formativo. Yan *et al.* (2025) utilizan ChatGPT como tutor en cursos de estructuras de datos con 180 estudiantes. En los resultados obtienen un tamaño de efecto d = 0.52 en pensamiento computacional; sobre el interés los estudiantes indican una mayor confianza en su aprendizaje cuando pueden experimentar libremente.

Chang (2025) muestra cómo en Taiwán el uso de ChatGPT necesita esfuerzos para mejorar la formación de los estudiantes en su uso, incluyendo su capacidad para juzgar el contenido generado. Se debe mejorar la compartición del conocimiento entre estudiantes, profesores y los sistemas de inteligencia artificial y los espacios de aprendizaje.

Un análisis de Kasneci et al. (2023) presenta los beneficios y riesgos del uso de ChatGPT en la educación. Entre los beneficios están la tutorización 24/7, la reducción de la ansiedad y la posibilidad de experimentar sin limitaciones. Entre los riesgos están la dependencia excesiva, las alucinaciones de código, el sesgo en las soluciones propuestas, la falta de depuración y la falta de validación por el estudiante.

Síntesis de evidencias: cinco componentes clave para la implementación

Tras el análisis anterior se identifican cinco componentes clave determinantes de efectividad:

1. Retroalimentación inmediata contextualizada: Debe ser inmediata (mismo día), específica (qué está mal y por qué), factible (cómo se mejora). Hay que evitar siempre la retroalimentación genérica.
2. Personalización según el nivel de conocimientos: Los estudiantes progresan por niveles (conceptual a sintáctico a arquitectónico). La estructura de aprendizaje debe ajustarse dinámicamente según su desarrollo.
3. Andamiaje cognitivo progresivo: Hay que reducir el apoyo cuando aumenta la competencia. Hay que conseguir un apoyo suficiente para que estudiante progrese, pero no tanto que impida el desarrollo de su autonomía.
4. Multimodalidad en la presentación de contenidos: Cada concepto se debe presentar en varios formatos (diagrama, código, visualización, vídeo). Se consigue reducir la carga cognitiva y permite que el estudiante use diferentes estilos de aprendizaje.
5. Analítica predictiva para realizar intervenciones tempranas: Hay que detectar patrones de trabajo (inactividad, tasa de error, dependencia de la IA) para poder predecir el abandono o que se produce un aprendizaje superficial.

Limitaciones y desafíos

En la revisión de estudios analizados también se indican limitaciones y desafíos que hay que tener en cuenta:

- Limitaciones técnicas: La dependencia excesiva de la IA que indica en Chang (2025) sugiere el riesgo real de que los estudiantes confundan conocimiento con comprensión. Ocurre todavía que el código generado sea sintácticamente correcto, pero algorítmicamente ineficiente o simplemente incorrecto.
- Limitaciones pedagógicas: Se observa una gran diversidad según el contexto institucional, la implementación concreta y las características de los estudiantes. No se puede aplicar la misma fórmula en todos los lugares y que funcione de la misma forma. Es necesaria una adaptación al entorno.
- Limitaciones de medición: Es necesario disponer de estándares de evaluación homogéneos. En cada estudio se indican formas diferentes de medir los resultados lo que complica obtener conclusiones realistas.

- Limitaciones éticas: Hay que poner un especial cuidado en la privacidad de los datos, el acceso a las tecnologías de IA, el posible sesgo algorítmico y la evaluación de la autenticidad de los resultados de los estudiantes.
- Dependencia cognitiva: Los estudiantes pueden obtener código funcional sin desarrollar una comprensión de los algoritmos subyacentes. Esta dependencia podría inhibir el desarrollo de habilidades fundamentales de resolución de problemas.
- Formación docente: El profesorado requiere no solo competencia técnica, sino conocimiento de estrategias pedagógicas específicas para la era de la IA.

Propuesta metodológica

A partir de las evidencias identificadas en la Sección 2 y considerando las limitaciones actuales, se propone un marco metodológico estructurado en cuatro fases secuenciales para la integración efectiva de IA generativa en la enseñanza universitaria de programación.

Fase 1: Diagnóstico y preparación

Esta fase establece la base para la intervención con IA, identificando competencias, recursos pedagógicos y tecnológicos. El proceso consiste en realizar un análisis de competencias iniciales, realizar un inventario de herramientas y formar al profesorado en las herramientas y técnicas de la IA.

Para el análisis de competencias iniciales es importante realizar pruebas a los estudiantes que permitan valorar entre otros aspectos, su capacidad de escribir código sin errores sintácticos, su capacidad para resolver problemas algorítmicos básicos, su capacidad para la descomposición de problemas complejos en subproblemas o la identificación y corrección de errores en código, entre otros. Con este análisis se puede detectar si existen grupos heterogéneos de estudiantes y establecer un marco base para las iniciativas posteriores.

Una vez valoradas las competencias del grupo, se puede establecer la línea inicial de trabajo usando también métricas que se pueden obtener del LMS como las calificaciones obtenidas por los distintos estudiantes, las ratios de aprobados, los patrones de uso del LMS, etc., que puedan aportar valor para el trabajo con el grupo.

En esta fase también hay que conseguir disponer de una relación de las herramientas y mecanismos de IA disponibles entre las que se pueden contar los sistemas de tutoría inteligente (ITS), los modelos de lenguaje como GitHub Copilot Education, ChatGPT API, Claude u otros y las plataformas de retroalimentación automática para programación como CodeGrade, GradeScope, etc., así como las herramientas de desarrollo y visualización. La relación de herramientas final debe ir acompañada de un plan de acceso que garantice que ningún estudiante queda en desventaja técnica por motivos económicos.

Por último, es importante que los docentes tengan la capacitación adecuada para, por una parte, utilizar las herramientas de IA de forma útil dentro del marco docente y, por otra, de crear una estructura formal con los estudiantes que les permita integrar dichas herramientas de IA con la estructura docente de las asignaturas. En este sentido los docentes deben disponer de formación en temas como el funcionamiento de los LLM, qué son las alucinaciones, cómo realizar *prompts* educativos efectivos, cómo interpretar los resultados de la analítica docente, cómo enseñar a integrar la IA en el proceso de validación, así como no menos importante, las consideraciones éticas, de privacidad y de protección de datos que pueden ser relevantes.

Fase 2: Diseño de estrategias personalizadas

Durante esta fase se diseñan las rutas de aprendizaje adaptativo, los niveles de andamiaje cognitivo y los flujos de interacción que se implantarán en la siguiente fase. Los niveles de andamiaje cognitivo se pueden estructurar en tres niveles progresivos que permite avanzar en el aprendizaje de los procesos de la programación de forma que el estudiante comprenda los conceptos desde lo más simple a lo más complejo, desde las estructuras sintácticas hasta las arquitecturas de desarrollo. Estos tres niveles serían:

- Nivel conceptual: se trabajan los paradigmas de programación, pudiendo ser sin código, utilizando pseudocódigo. En este nivel lo relevante es trabajar los conceptos básicos de forma explicativa.
- Nivel sintáctico: se trabaja la transformación del nivel conceptual en un nivel básico de programación donde es importante llegar a entender la forma de construcción de las estructuras de programación, siendo un recurso muy valioso la corrección de errores y la compleción de código a partir de esqueletos de este.
- Nivel arquitectónico: se trabaja la construcción de programas a partir de las estructuras de programación para crear soluciones completas que integren los conceptos ya conocidos. Es necesario trabajar la validación de la solución como forma de dar respuesta a un problema planteado inicialmente.

Para estas fases, se pueden diseñar estrategias de interacción basadas en técnicas formales de ingeniería de instrucciones (*Prompt engineering*). En la literatura se sugiere estructurar las consultas mediante patrones específicos para reducir las alucinaciones y mejorar el razonamiento pedagógico del modelo.

- Nivel 1: Razonamiento analógico (*Zero-Shot Analogical Reasoning*). Se utilizan *prompts* que indican que actúe como un tutor, prohibiendo explícitamente la generación de código y forzando el uso de ejemplos de la vida real para asentar el modelo mental del estudiante.
- Nivel 2: Cadena de pensamiento (*Chain-of-Thought - CoT*). Para la depuración sintáctica, se indica que desglose el razonamiento paso a paso antes de ofrecer una solución. Esta técnica permite al estudiante visualizar la "traza de ejecución" lógica del algoritmo, facilitando la identificación de errores lógicos.

- Nivel 3: Aprendizaje con pocos ejemplos (*Few-Shot Learning*) y contraste. Se proporcionan ejemplos de código funcional pero ineficiente frente a código optimizado, solicitando un análisis comparativo. Se aplican restricciones (*Constraint-based prompting*) para forzar la optimización de complejidad temporal o espacial.

Ejemplo: Concepto "Bucles for"

Nivel 1 (Conceptual - Socratic & Analogical): "Actúa como profesor de programación para principiantes. Explica el concepto de 'bucle' sin utilizar código. Utiliza ejemplos de la vida cotidiana para la repetición controlada y haz una pregunta final para verificar si lo he entendido."

Nivel 2 (Sintáctico - Chain-of-Thought Debugging): "Mi bucle no realiza la suma esperada. No me des la solución corregida. Analiza mi código paso a paso, simulando el valor de la variable 'suma' en cada repetición e identifica dónde difiere mi lógica de la ejecución real."

Nivel 3 (Arquitectónico - Comparative Analysis & Constraints): "Te doy una solución funcional con bucles con complejidad $O(n^2)$. Escribe dos soluciones diferentes: una priorizando la legibilidad y otra priorizando el rendimiento a $O(n)$. Compara las soluciones y explica las ventajas y desventajas de cada una."

Como medida para asentar las competencias en los tres niveles mencionados es relevante incorporar la multimodalidad. Para cada concepto fundamental se puede incorporar diversas modalidades de presentación de los contenidos como pueden ser diagramas de flujo que permitan visualizar paso a paso la ejecución de código, por ejemplo, mostrando la pila de llamadas, la visualización dinámica del estado de la memoria durante la ejecución, la representación de distintas soluciones de un mismo problema para entender los conceptos de complejidad, etc. De esta forma, se consigue reducir la carga cognitiva al presentar información en múltiples canales, permitiendo al estudiante adquirir las competencias mediante la representación que mejor se adapta a su estilo de aprendizaje.

Otro aspecto relevante de las estrategias es cómo ofrecer retroalimentación. La retroalimentación es uno de los factores que afectan de forma significativa en los procesos de aprendizaje. Todas las estrategias de retroalimentación deberían incluir qué se detecta que no es correcto, por qué se detecta que no es correcto, cómo se puede abordar la corrección del problema detectado y cómo realizarlo haciendo siempre referencia al concepto subyacente.

Fase 3: Implementación y monitoreo

En esta fase se despliegan las estrategias en el entorno formativo y se realiza un seguimiento de los estudiantes, capturando los datos de interacción de forma que se puedan tomar medidas de intervención cuando se detecten patrones de riesgo.

En esta fase se trata de crear situaciones que puedan desplegar el diseño de las estrategias planteadas en la fase anterior en el proceso formativo. En este momento se utilizan las herramientas y las estrategias las usan dentro de un entorno de aprendizaje como Moodle, Canvas, etc. de manera que el estudiante pueda acceder a los ejercicios que le correspondan con el nivel de andamiaje cognitivo detectado y que se le asigna de acuerdo con las interacciones que ya haya tenido anteriormente, la retroalimentación que se genera para su nivel y los problemas que se detecten de la actuación del estudiante.

Durante este proceso de trabajo con el estudiante se recogen los datos de interacción con el sistema. De esta forma se puede disponer de información sobre los patrones de interacción y aprendizaje de los estudiantes a través de datos como el tiempo de resolución de los ejercicios, el número de compilaciones/ejecuciones realizadas, la frecuencia y tipo de los errores, el uso de recursos de ayuda como los foros, documentación, etc. y las consultas realizadas a la IA valorando qué preguntó, si el estudiante aceptó la sugerencia, etc.

En este entorno se pueden aplicar técnicas de *machine learning* para identificar estudiantes en riesgo antes de que se conviertan en un problema. Entre los indicadores que se pueden tener en cuenta para la detección temprana del riesgo pueden estar la inactividad prolongada en el sistema, la alta tasa de error en los ejercicios, la dependencia excesiva con respecto de la IA o incluso los tiempos anormalmente pequeños de resolución, lo que puede suponer copia de IA sin comprensión.

En el momento en que se detecta la posibilidad de riesgo del estudiante cuando se pueden iniciar las medidas correctivas con el estudiante de forma que se ponga en alerta al docente con una descripción de las causas detectadas. A partir de los datos recogidos incluso se podrían dar sugerencias concretas de intervención con el estudiante con objeto de ofrecer, tutorías, materiales especiales o determinado tipo de andamiaje cognitivo.

Fase 4: Evaluación y mejora continua

En esta fase se realiza una evaluación de los resultados y la generación de retroalimentación para ajustar la metodología. Para ello, se comparan las métricas pre y post intervención valorando los tamaños de efecto en calificaciones y tiempos de resolución para cuantificar el impacto de la IA en competencias de programación.

Para realizar la valoración de los resultados se pueden utilizar distintos instrumentos de evaluación multidimensionales. Entre los instrumentos cuantitativos pueden estar las pruebas de conocimiento, las métricas de rendimiento como las calificaciones finales, la tasa de aprobados, el promedio de intentos por ejercicio, etc., el análisis de patrones de uso de IA como el tiempo promedio en tareas, frecuencia de errores, etc., o el uso de cuestionarios validados. Entre los instrumentos cualitativos se pueden emplear grupos focales con estudiantes representativos, entrevistas con docentes sobre la experiencia de implantación, el uso de encuestas de satisfacción y la observación sobre cómo los estudiantes usan las herramientas y qué preguntas realizan.

Con esta información se puede realizar un ciclo de mejora continua (Plan, Do, Check, Act) de forma que Plan (Basado en la evaluación, se identifican áreas de mejora específicas), Do (Se implementan cambios en pequeña escala) Check (Se evalúa el impacto de los cambios) y Act (se escalan los cambios que hayan resultado beneficiosos).

Conclusiones

En este artículo se ha realizado una propuesta metodológica para la integración de la IA generativa en la enseñanza universitaria de programación. Tras una revisión de artículos actualizada, se identificaron cuatro áreas fundamentales, la tutoría inteligente, la retroalimentación automatizada, la personalización adaptativa y el uso de LLM. Se concretan en cinco componentes para su implementación efectiva: retroalimentación inmediata contextualizada, personalización según nivel de conocimientos, andamiaje cognitivo progresivo, multimodalidad en presentación de contenidos, y analítica predictiva para intervenciones tempranas.

La principal contribución del trabajo es una propuesta metodológica en cuatro fases que integra coherentemente las evidencias sobre IA en educación de programación. A diferencia de propuestas previas que se centran en herramientas concretas o en intervenciones puntuales, esta propuesta proporciona un proceso sistemático que cubre desde el diagnóstico inicial hasta la mejora continua, una integración explícita de los cinco componentes clave identificados en evidencia empírica, un equilibrio entre la automatización y la supervisión humana para reducir los riesgos de dependencia excesiva de la IA, mecanismos de personalización adaptativa basados en analítica y analítica predictiva para la intervención.

Las implicaciones de esta propuesta resultan importantes tanto para docentes de programación a los que proporciona una guía para incorporar la IA, facilitando la transición a entornos con IA fundamentada en evidencia, a las instituciones educativas con un modelo que orienta las decisiones sobre tecnologías educativas, priorizando los recursos a intervenciones con evidencia de efectividad. La propuesta de cuatro fases permite una adopción gradual, lo que reduce riesgos financieros y pedagógicos. Además, para los estudiantes se busca maximizar los beneficios de la IA a la vez que desarrollar habilidades de validación, evaluación y uso ético de herramientas de la IA, frente a desarrollar dependencia.

Hay que considerar también que la propuesta presenta limitaciones importantes. Requiere una inversión inicial en formación docente y la contratación de licencias de IA y otras herramientas, lo que puede suponer un problema para algunas instituciones. Además, para aplicar los algoritmos de analítica predictiva se necesitan datos históricos suficientes para que sean fiables, lo que complica la implantación en cursos nuevos. La propuesta requiere todavía una validación mediante estudios experimentales controlados en contextos diversos. Aunque nuestra experiencia actual es positiva, no es aún concluyente.

La integración de la IA generativa en la formación de programación universitaria requiere de más que una adopción tecnológica. Es necesario un enfoque sistemático que se fundamente en evidencias pedagógicas y se centre en el desarrollo de la autonomía del estudiante. La propuesta realizada representa un primer paso hacia la integración coherente de la IA, utilizando una síntesis de evidencias actuales.

Referencias

Callejo Pinardo, P., Alario Hoyos, C., & Delgado Kloos, C. (2024). *Evaluating the Impact of ChatGPT on Programming Learning Outcomes in a Big Data Course*. https://e-archivo.uc3m.es/entities/publication/1f2e97cc-7882-41e8-b7a9-01083d783cce

Chang, H.-P. (2025). Research on the Application of Generative Artificial Intelligence in the Field of Design Teaching at a Taiwan University of Science. *Frontier Computing: Volume 4: Proceedings of FC 2024*, *1358*, 240.

Hattie, J., & Timperley, H. (2007). The Power of Feedback. *Review of Educational Research*, *77*(1), 81-112. https://doi.org/10.3102/003465430298487

Huang, X., Xu, W., & Liu, R. (2025). Effects of Intelligent Tutoring Systems on Educational Outcomes: Evidence From a Comprehensive Analysis. *International Journal of Distance Education Technologies (IJDET)*, *23*(1), 1-25.

Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., … Hüllermeier, E. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences*, *103*, 102274.

Mayer, R. E. (2024). The Past, Present, and Future of the Cognitive Theory of Multimedia Learning. *Educational Psychology Review*, *36*(1), 8. https://doi.org/10.1007/s10648-023-09842-1

Paiva, J. C., Leal, J. P., & Figueira, Á. (2022). Automated Assessment in Computer Science Education: A State-of-the-Art Review. *ACM Transactions on Computing Education*, *22*(3), 1-40. https://doi.org/10.1145/3513140

Pitts, G., Hridi, A. P., & Narayanan, A. B. L. (2025). A Survey of LLM-Based Applications in Programming Education: Balancing Automation and Human Oversight. *Proceedings of the Fourth Workshop on Bridging Human-Computer Interaction and Natural Language Processing (HCI+ NLP)*, 255-262. https://aclanthology.org/2025.hcinlp-1.21/

Reihanian, I., Hou, Y., Chen, Y., & Zheng, Y. (2025). A Review of Generative AI in Computer Science Education: Challenges and Opportunities in Accuracy, Authenticity, and Assessment. En H. R. Arabnia, L. Deligiannidis, F. Shenavarmasouleh, S. Amirian, & F. Ghareh Mohammadi (Eds.), *Computational Science and Computational Intelligence* (pp. 144-158). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-94943-2_11

St-Hilaire, F., Vu, D. D., Frau, A., Burns, N., Faraji, F., Potochny, J., … Glazier, T. (2022). A new era: Intelligent tutoring systems will transform online learning for millions. *arXiv preprint arXiv:2203.03724*. https://arxiv.org/abs/2203.03724

Sweller, J. (2020). Cognitive load theory and educational technology. *Educational Technology Research and Development*, *68*(1), 1-16. https://doi.org/10.1007/s11423-019-09701-3

Tan, S., Rudolph, J., & Tan, S. (2024). Riding the Generative AI Tsunami: Addressing the Teaching and Learning Crisis in Higher Education. En J. Rudolph, J. Crawford, C.-Y. Sam, & S. Tan (Eds.), *The Palgrave Handbook of Crisis Leadership in Higher Education* (pp. 135-154). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-54509-2_8

Wisniewski, B., Zierer, K., & Hattie, J. (2020). The power of feedback revisited: A meta-analysis of educational feedback research. *Frontiers in psychology*, *10*, 487662.

Yan, Y.-M., Chen, C.-Q., Hu, Y.-B., & Ye, X.-D. (2025). LLM-based collaborative programming: Impact on students' computational thinking and self-efficacy. *Humanities and Social Sciences Communications*, *12*(1), 1-12.

Zhang, J., Jantakoon, T., & Laoha, R. (2025). Meta-Analysis of Artificial Intelligence in Education. *Higher Education Studies*, *15*(2), 189-210.

Dr. Jesús Sánchez Allende actualmente es profesor en CUNEF Universidad y posee más de veinte años de experiencia como profesor universitario. Ha desarrollado múltiples planes de estudio de nivel de grado y máster. Ha liderado múltiples proyectos de transformación e innovación educativa en niveles universitarios sobre el uso de nuevas tecnologías para mejorar la formación basada en la evidencia. Entre sus trabajos se encuentran la personalización adaptativa y el análisis predictivo en educación fundamentalmente en el área de la programación. Ha publicado varios libros relacionados con la programación, así como artículos académicos que aplican las nuevas tecnologías en el ámbito de formación universitaria.