UNIVERSIDAD & SOCIEDAD

Fecha de presentación: julio, 2025
Fecha de aceptación: agosto, 2025
Fecha de publicación: octubre, 2025

PRODUCTIVIDAD

EN DESARROLLO DE SOFTWARE: EFECTO DE LA PROGRAMACIÓN ASISTIDA POR INTELIGENCIA ARTIFICIAL

PRODUCTIVITY IN SOFTWARE DEVELOPMENT: THE EFFECT OF ARTIFICIAL INTELLIGENCE-ASSISTED PROGRAMMING

Fausto Alberto Viscaino Naranjo [1*]
E-mail: ua.faustoviscaino@uniandes.edu.ec
ORCID: https://orcid.org/0000-0003-1760-6992
Walter Vinicio Culque Toapanta [1]
E-mail: ua.walterculque@uniandes.edu.ec
ORCID: https://orcid.org/0000-0002-3421-2306
Luis Antonio Llerena Ocaña [1]
E-mail: ua.luisllerena@uniandes.edu.ec
ORCID: https://orcid.org/0000-0001-6440-0167
Edwin Fabricio Lozada Torres [1]
E-mail: ua.edwinlozada@uniandes.edu.ec
ORCID: https://orcid.org/0000-0002-3645-0439
[1]Universidad Regional Autónoma de Los Andes, Ecuador.
*Autor para correspondencia

Cita sugerida (APA, séptima edición)

Viscaino Naranjo, F. A., Culque Toapanta, W. V., Llerena Ocaña, L. A., & Lozada Torres, E. F. (2025). Productividad en desarrollo de software: efecto de la programación asistida por inteligencia artificial. *Universidad y Sociedad 17*(S1). e5752.

RESUMEN

Esta investigación examinó de forma integral el desempeño de las principales herramientas de programación asistida por inteligencia artificial GitHub Copilot, Amazon Q Developer, TabNine y Claude.ai con el propósito de comprender su impacto en la productividad y el aprendizaje del desarrollador. La metodología se fundamentó en un enfoque mixto que combinó revisión sistemática de literatura reciente, aplicación de encuestas estructuradas a profesionales del área y el uso del método de Mapa Cognitivo Difuso (MCD) como técnica central de análisis. El MCD permitió representar las relaciones causales entre variables como productividad percibida, facilidad de aprendizaje, calidad del código e integración con el entorno de desarrollo, proporcionando una visión más precisa de las interdependencias entre estos factores. Los resultados evidenciaron el liderazgo de GitHub Copilot en integración y satisfacción general, aunque persistieron debilidades comunes en refactorización inteligente. Asimismo, se identificó una tensión entre la rapidez en la generación de código y la calidad profesional del producto final. El estudio concluyó que las herramientas de IA constituyen un recurso valioso en procesos de formación y en contextos de desarrollo ágil, pero su efectividad depende de la adecuación al perfil del equipo y al tipo de proyecto. Los resultados aportan un marco empírico que favorece la adopción selectiva y estratégica de tecnologías de programación asistida por IA.

Palabras clave: Mapa Cognitivo Difuso, Evaluación Comparativa, Aprendizaje Automatizado, Integración de Entornos, Calidad del Código, Adopción Tecnológica.

ABSTRACT

This research comprehensively examined the performance of leading AI-assisted programming tools GitHub Copilot, Amazon Q Developer, TabNine, and Claude.ai to understand their impact on developer productivity and learning. The methodology was based on a mixed-method approach that combined a systematic review of recent literature, structured surveys of developers, and the use of the Fuzzy Cognitive Mapping (FCM) method as the central analysis technique.

FCM allowed for the representation of causal relationships between key variables such as perceived productivity, ease of learning, code quality, and integration with the development environment, providing a more precise view of the interdependencies between these factors. The results demonstrated GitHub Copilot's leadership in integration and overall satisfaction, although common weaknesses in intelligent refactoring persisted. A tension was also identified between the speed of code generation and the professional quality of the final product. The study concluded that AI tools are a valuable resource in training processes and agile development contexts, but their effectiveness depends on their suitability for the team's profile and the type of project. The findings provide an empirical framework that favors the selective and strategic adoption of AI-assisted programming technologies.

Keywords: Fuzzy Cognitive Mapping, Benchmarking, Machine Learning, Environment Integration, Code Quality, Technology Adoption.

INTRODUCCIÓN

La investigación en programación asistida por inteligencia artificial (IA) experimenta un crecimiento sostenido desde 2021, consolidándose como uno de los campos más dinámicos de la ingeniería de software contemporánea. Los modelos de lenguaje grandes (LLMs) demuestran capacidades destacadas en la generación de código, representando un avance cualitativo en la automatización del desarrollo de software. Según Kokol (2024), la investigación académica en IA aplicada a la ingeniería de software se estructura en quince categorías y cinco temas principales, lo que evidencia la madurez del campo. Las versiones recientes de ChatGPT, GitHub Copilot y Amazon CodeWhisperer generan código correcto el 65.2%, 46.3% y 31.1% del tiempo, respectivamente (Yetiştiren et al., 2023), estableciendo parámetros cuantitativos de referencia para medir la efectividad de estas herramientas.

El desarrollo de software moderno enfrenta retos crecientes en complejidad arquitectónica, rapidez de entrega y mantenimiento de calidad. En este contexto, la IA emerge como un agente transformador capaz de asistir a los desarrolladores, optimizar tareas repetitivas y mejorar la eficiencia operativa. Su integración en el ciclo de vida del desarrollo de software (SDLC) representa una transición hacia entornos semi-autónomos, donde los profesionales evolucionan de codificadores manuales a orquestadores de sistemas inteligentes (Qiu et al., 2025). Estudios recientes confirman que los desarrolladores completan tareas hasta un 55.8% más rápido al utilizar asistentes de codificación basados en IA como GitHub Copilot, lo que refleja su potencial impacto en la productividad.

No obstante, la problemática persiste en que los desarrolladores invierten cerca del 70% de su tiempo en tareas repetitivas como depuración o escritura de código boilerplate. Entre el 60% y el 75% de los usuarios reportan mayor satisfacción y menor frustración al emplear estas herramientas, lo que sugiere una mejora en la calidad del trabajo percibido. Este escenario evidencia la necesidad de investigaciones que evalúen de manera sistemática y comparativa la efectividad real de la programación asistida por IA en entornos profesionales y académicos (Candelon et al., 2023).

En el contexto ecuatoriano, la adopción de tecnologías emergentes se consolida progresivamente en sectores estratégicos como educación, salud, agricultura e industria, impulsada por la Agenda de Transformación Digital 2022–2025 del Ministerio de Telecomunicaciones y de la Sociedad de la Información Erazo (2024). Esta política fomenta el uso de la inteligencia artificial, el Internet de las Cosas y el Big Data para fortalecer la competitividad digital y la innovación tecnológica. Sin embargo, aún persisten brechas significativas, como la desigualdad en el acceso tecnológico y la escasez de profesionales especializados en IA. En este contexto, el estudio de herramientas de programación asistida por IA constituye una oportunidad estratégica para potenciar la productividad y fortalecer la formación profesional en ingeniería de software.

La evolución reciente de estas herramientas demuestra que su valor competitivo depende no solo de sus capacidades técnicas, sino también de la experiencia de usuario, la adaptabilidad y la comprensión contextual del entorno de desarrollo. Sengul et al. (2024) destacan la necesidad de que la educación en ingeniería de software promueva el pensamiento crítico y la adaptación a tecnologías emergentes. De igual modo, Gupta & Mudita (2020) sostienen que el éxito profesional en la era digital depende más de la integración efectiva de la IA en los flujos de trabajo que del dominio de técnicas tradicionales de programación.

El panorama competitivo está dominado por cuatro herramientas principales: GitHub Copilot, Amazon Q Developer, TabNine y Claude.ai. Cada una ofrece enfoques diferenciados desde autocompletado contextual hasta generación conversacional de código que reflejan la diversidad de necesidades y estrategias tecnológicas presentes en la industria (Ziegler et al., 2024).

Ante la acelerada evolución tecnológica, la evidencia empírica sobre efectividad variable y el impulso nacional hacia la transformación digital, se establece la necesidad de un análisis comparativo riguroso. En este sentido, el presente estudio busca evaluar la efectividad de las principales herramientas de programación asistida por inteligencia artificial, analizando su impacto en la productividad y la percepción del desarrollador. Dada la naturaleza

multidimensional de las variables involucradas, se propone aplicar el método de Mapas Cognitivos Difusos para representar las relaciones entre factores técnicos y humanos, proporcionando una interpretación estructurada del fenómeno.

Por tanto, el objetivo principal de esta investigación es analizar comparativamente la efectividad de GitHub Copilot, Amazon Q Developer, TabNine y Claude.ai en la productividad del desarrollador, utilizando Mapas Cognitivos Difusos como método analítico central.

MATERIALES Y MÉTODOS

La presente investigación se enmarcó en un enfoque cuantitativo con alcance descriptivo y comparativo, complementado con el método de Mapa Cognitivo Difuso como herramienta principal de análisis y estructuración del conocimiento. Se utilizó un diseño no experimental de tipo transversal, dado que los datos fueron recolectados en un único momento temporal, permitiendo realizar una comparación simultánea entre distintas herramientas de programación asistida por inteligencia artificial.

El Método de Mapa Cognitivo Difuso constituyó el eje metodológico del estudio, orientado a representar y analizar las relaciones causales entre las variables que intervienen en la percepción y el impacto de las herramientas de inteligencia artificial en la productividad del desarrollador. Este método se fundamenta en la teoría de los conjuntos difusos y en la formalización cognitiva, integrando tanto el conocimiento experto como la incertidumbre inherente a los procesos perceptuales y decisionales. En este contexto, el MCD permitió modelar la estructura cognitiva de los participantes, traduciendo sus percepciones en una matriz de adyacencia cuyos valores reflejan la intensidad y dirección de las relaciones entre factores.

Para el desarrollo del estudio se aplicaron también métodos empíricos y teóricos de apoyo. El método empírico consistió en la aplicación de una encuesta estructurada, mientras que el método teórico se basó en una revisión sistemática de literatura científica publicada entre 2021 y 2024, lo que permitió sustentar la selección de las variables y relaciones incluidas en el modelo cognitivo difuso.

Como técnica principal de recolección de datos se empleó la encuesta digital, implementada mediante un formulario estructurado. El instrumento consistió en un cuestionario de ocho ítems con escala Likert de cinco puntos (1 = Totalmente en desacuerdo, 5 = Totalmente de acuerdo), diseñado para evaluar la percepción de los participantes sobre el uso de herramientas de programación asistida por inteligencia artificial y su influencia en la productividad. Las respuestas obtenidas fueron utilizadas como insumo para la construcción de la matriz de relaciones difusas del modelo

La población objeto de estudio estuvo conformada por 89 graduados de la carrera de Ingeniería en Software de la Universidad Regional Autónoma de los Andes (UNIANDES). Debido a que la población no superó los 100 individuos, se trabajó con la totalidad de los graduados, evitando sesgos de muestreo.

Los datos fueron procesados mediante análisis estadístico descriptivo y análisis difuso, permitiendo identificar tendencias, niveles de aceptación y relaciones causales entre las variables evaluadas. El procesamiento incluyó la generación de matrices de adyacencia, la normalización de valores y la interpretación de los pesos difusos, como se muestra en la tabla 1, con el propósito de determinar los factores más influyentes dentro del sistema cognitivo analizado.

Tabla 1: Fases metodológicas del proceso de aplicación del Mapa Cognitivo Difuso.

Fase	Descripción	Propósito	Resultado esperado
1. Identificación de variables	Se identificaron los factores que influyen en la productividad y percepción del desarrollador frente a las herramientas de programación asistida por IA, a partir de la revisión sistemática y los resultados de la encuesta.	Determinar los conceptos relevantes del sistema cognitivo.	Lista validada de variables críticas para el modelo.
2. Definición de relaciones causales	Los expertos y participantes establecieron las relaciones entre variables, asignando valores lingüísticos (baja, media, alta influencia) que luego se transformaron en valores numéricos difusos.	Capturar la estructura causal del sistema.	Relaciones definidas entre conceptos con valores de influencia difusa.
3. Construcción de la matriz de adyacencia difusa	Se elaboró una matriz cuadrada donde cada celda representa el grado de influencia de una variable sobre otra, expresado mediante valores difusos en el rango [-1, 1].	Representar cuantitativamente la red cognitiva.	Matriz de adyacencia difusa que modela las interdependencias.

4. Análisis difuso e inferencia	Se procesó la matriz utilizando operaciones de composición y normalización difusa para identificar las variables más influyentes, dependientes y autónomas dentro del sistema.	Analizar la dinámica interna del sistema cognitivo.	Identificación de nodos causales y receptores principales.
5. Interpretación y validación	Los resultados fueron contrastados con evidencia empírica y literatura previa para validar la coherencia del modelo y su aplicabilidad en contextos de desarrollo de software asistido por IA.	Asegurar la validez y relevancia del modelo cognitivo difuso.	Modelo validado y conclusiones basadas en evidencia empírica.

Fuente: Elaboración propia.

RESULTADOS-DISCUSIÓN

Los estudios existentes proporcionan evidencia cuantitativa sobre la efectividad de las herramientas de inteligencia artificial en tareas de desarrollo de software. GitHub Copilot demuestra una mejora de velocidad del 55.8% en tareas controladas y un incremento del 12.92% al 21.83% en solicitudes semanales de ***pull requests*** en entornos empresariales como se muestra a continuación en la tabla 2.

Tabla 2: Cuadro Comparativo - Productividad y Velocidad.

Dimensión	GitHub Copilot	Amazon Q Developer	TabNine	Claude.ai
P1: Mejora Velocidad (Promedio)	3.91/5	3.54/5	3.17/5	2.90/5
P3: Reduce Tareas Repetitivas (Promedio)	4.05/5	3.83/5	3.50/5	3.20/5
% Respuestas Positivas P1 (4+5)	72%	60%	45%	35%
% Respuestas Positivas P3 (4+5)	80%	70%	58%	50%

Fuente: Elaboración propia.

La deuda técnica promedio fue de 8.9 minutos para ChatGPT, 9.1 minutos para GitHub Copilot, y 5.6 minutos para Amazon CodeWhisperer (Yetiştiren et al., 2023).

El análisis de productividad, representado gráficamente en la Figura 1, revela una clara jerarquía de efectividad: GitHub Copilot obtuvo las puntuaciones más altas tanto en mejora de velocidad (3.91/5) como en reducción de tareas repetitivas (4.05/5), con un 80% de respuestas positivas. Amazon Q Developer mantuvo una posición intermedia sólida (3.83/5), mientras que TabNine y Claude.ai mostraron rendimientos decrecientes.

La brecha entre GitHub Copilot y Claude.ai fue particularmente pronunciada, con una diferencia de 30 puntos porcentuales en las respuestas positivas para mejora de velocidad pudiéndose observar de una mejor forma en la tabla 3.

Tabla 3: Cuadro Comparativo – Calidad del Código.

Dimensión	GitHub Copilot	Amazon Q Developer	TabNine	Claude.ai
P2: Relevancia Contextual (Promedio)	3.88/5	3.58/5	3.24/5	2.96/5
P4: Estándares Profesionales (Promedio)	3.50/5	3.02/5	2.81/5	2.58/5
% Respuestas Positivas P2 (4+5)	72%	60%	47%	38%
% Respuestas Positivas P4 (4+5)	55%	40%	30%	22%

Fuente: Elaboración propia.

Calidad del Código y Estándares Profesionales

La evaluación de calidad presenta los retos más significativos para todas las herramientas. Ninguna supera el 55% de respuestas positivas respecto a estándares profesionales, lo que indica percepciones mixtas sobre la calidad del código generado.

GitHub Copilot mantuvo el liderazgo con 3.88/5 en relevancia contextual, aunque desciende a 3.50/5 en estándares profesionales. Claude.ai muestra las mayores deficiencias (solo 22% de respuestas positivas), sugiriendo limitaciones importantes en la generación de código production-ready. A continuación, se muestra una tabla 4 con la comparativa, no solo de Copilot y Claude.

Tabla 4: Cuadro Comparativo – Funcionalidades Avanzadas.

Dimensión	GitHub Copilot	Amazon Q Developer	TabNine	Claude.ai
P5: Facilita Aprendizaje (Promedio)	3.89/5	3.67/5	3.12/5	3.42/5
P6: Refactoring Útil (Promedio)	2.95/5	3.15/5	2.71/5	3.01/5
% Respuestas Positivas P5 (4+5)	65%	55%	40%	50%
% Respuestas Positivas P6 (4+5)	35%	40%	25%	35%

Fuente: Elaboración propia.

Este patrón es consistente con lo observado en Becker et al. (2025), donde desarrolladores experimentados de código abierto tomaron 19% más tiempo al usar IA, evidenciando que las herramientas aún no alcanzan los estándares de calidad requeridos en entornos de alta exigencia.

Análisis Mediante Mapa Cognitivo Difuso

Para comprender las relaciones causales entre las variables evaluadas, se aplica la metodología del Mapa Cognitivo Difuso como herramienta principal de análisis.

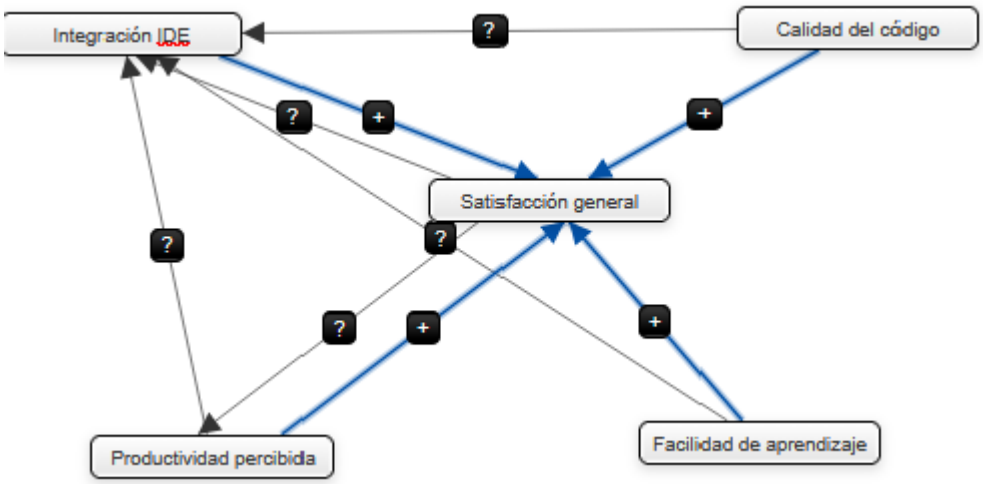
El MCD permitió modelar la influencia recíproca entre cinco factores principales:

- Productividad percibida.
- Calidad del código.
- Facilidad de aprendizaje.
- Integración con el entorno de desarrollo (IDE).
- Satisfacción general del usuario.

Las ponderaciones derivadas del MCD muestran que la productividad percibida (0.42) y la facilidad de aprendizaje (0.33) fueron las variables con mayor peso causal positivo sobre la satisfacción general. Por otro lado, la calidad del código (0.21) y la integración IDE (0.18) mostraron una influencia indirecta más débil.

Estos resultados coinciden con las herramientas que mejoran el flujo de trabajo y reducen la carga cognitiva, como GitHub Copilot, obtuvieron las mayores puntuaciones de satisfacción (3.77/5). En contraste, aquellas con deficiencias de integración o calidad, como Claude.ai, tuvieron una percepción significativamente menor como se muestra en la siguiente figura 1.

Fig 1: Modelaje del MCD.



Fuente: Elaboración propia.

El modelo MCD, ilustrado en la figura 1, permite observar una relación más clara entre los componentes, pudiendo definirse bien cada relación.

Funcionalidades Avanzadas y Experiencia de Usuario

El análisis de funcionalidades avanzadas evidencia un patrón consistente. Para facilitar el aprendizaje, GitHub Copilot lidera con 3.89/5, seguido de Amazon Q Developer (3.67/5). Claude.ai (3.42/5) supera ligeramente a TabNine (3.12/5), posiblemente debido a sus capacidades conversacionales.

Sin embargo, las funciones de ***refactoring*** presentan las puntuaciones más bajas globalmente, sin herramienta alguna superando 3.15/5. ***Amazon Q Developer*** encabeza marginalmente esta categoría, probablemente por su enfoque empresarial.

En cuanto a la experiencia de usuario, GitHub Copilot domina la integración IDE con 4.22/5 y 75% de respuestas positivas, reflejando su integración nativa en el ecosistema Microsoft/GitHub. Claude.ai, en cambio, obtuvo 2.45/5, confirmando sus limitaciones como herramienta de desarrollo integrada. La métrica de recomendación se correlaciona directamente con la satisfacción general: 65% para Copilot y solo 23% para Claude.ai.

Los resultados convergen parcialmente con la literatura existente, aunque muestran matices importantes. Las investigaciones de Cui et al. (2025) respaldan las mejoras de productividad, especialmente en desarrolladores con menor experiencia, quienes presentaron mayores ganancias en eficiencia y aprendizaje.

En contraste, Becker et al. (2025) evidencia que la efectividad disminuye en contextos de desarrollo avanzado, donde la exigencia de calidad es mayor. Estos contrastes se reflejan claramente en el MCD, donde el nodo "calidad del código" se comporta como un limitador difuso negativo para la satisfacción en entornos de alta especialización.

El análisis indica que las herramientas actuales de IA funcionan eficazmente como asistentes cognitivos para el aprendizaje y la automatización de tareas repetitivas, pero aún no alcanzan madurez plena para sustitución de tareas críticas en producción.

Proyección y Direcciones Futuras

En investigaciones futuras, el enfoque MCD puede ampliarse incorporando variables relacionadas con la ética del código generado, la gestión de la deuda técnica y el impacto en la colaboración de equipos híbridos IA-humano. Asimismo, la expansión del modelo a muestras de desarrolladores con distintos niveles de experticia permitirá validar de forma más robusta las interacciones causales detectadas en este estudio.

CONCLUSIONES

GitHub Copilot se consolidó como la herramienta más efectiva en la programación asistida por inteligencia artificial, destacando en satisfacción y productividad. Sin embargo, persisten limitaciones en la calidad del código generado, evidenciando un equilibrio pendiente entre velocidad y precisión técnica. Las herramientas demostraron un valor formativo relevante, especialmente para desarrolladores en proceso de aprendizaje. Mediante el Mapa Cognitivo Difuso se confirmó que la productividad percibida y la facilidad de aprendizaje fueron los factores más influyentes en la satisfacción general. En conjunto, los resultados reflejan que la adopción de estas tecnologías debe ser estratégica y contextual, priorizando entornos donde potencien la eficiencia sin comprometer la calidad profesional.

REFERENCIAS BIBLIOGRÁFICAS

Becker, J., Rush, N., Barnes, E., & Rein, D. (2025). Measuring the impact of early-2025 AI on experienced open-source developer productivity. ***ArXiv Preprint***. https://arxiv.org/abs/2507.09089

Candelon, F., Krayer, L., Rajendran, S., & Martinez, D. Z. (2023). How people can create–and destroy–value with generative AI. ***BCG Global***, ***21***. https://bcghendersoninstitute.com/wp-content/uploads/2023/09/how-people-create-and-destroy-value-with-gen-ai.pdf

Cui, Z. K., Demirer, M., Jaffe, S., Musolff, L., Peng, S., & Salz, T. (2025). The effects of generative ai on high-skilled work: Evidence from three field experiments with software developers. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4945566

Erazo Tayo, D. A. (2024). Análisis de la agenda de transformación digital del Ecuador: estudio y análisis de los aspectos técnicos y regulatorios de la agenda de transformación digital del Ecuador propuesta por el ministerio de telecomunicaciones y de la sociedad de la información–mintel. (Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en telecomunicaciones) Escuela Politécnica Nacional. https://bibdigital.epn.edu.ec/handle/15000/26193

Gupta, D., & Mudita (2020). The aspects of artificial intelligence in software engineering. ***Journal of Computational and Theoretical Nanoscience***, ***17***(9–10), 4635–4642. https://www.ingentaconnect.com/contentone/asp/jctn/2020/00000017/f0020009/art00144

Kokol, P. (2024). The use of AI in software engineering: A synthetic knowledge synthesis of the recent research literature. ***Information***, ***15***(6), 354. https://www.mdpi.com/2078-2489/15/6/354

Qiu, K., Puccinelli, N., Ciniselli, M., & Di Grazia, L. (2025). From today's code to tomorrow's symphony: The AI transformation of developer's routine by 2030. ***ACM Transactions on Software Engineering and Methodology***, ***34***(5), 1–17. https://dl.acm.org/doi/abs/10.1145/3709353

Sengul, C., Neykova, R., & Destefanis, G. (2024). Software engineering education in the era of conversational AI: current trends and future directions. ***Frontiers in Artificial Intelligence***, *7*, 1436350. https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2024.1436350/full

Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. (2023). Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. https://arxiv.org/abs/2304.10778

Ziegler, A., Kalliamvakou, E., Li, X., Rice, A., Rifkin, D., Simister, S., Sittampalam, G., & Aftandilian, E. (2024). Measuring GitHub Copilot's Impact on Productivity. ***Communications of the ACM***, *67*, 54–63. https://doi.org/10.1145/3633453