PRISMA ODS
REVISTA MULTIDISCIPLINARIA SOBRE DESARROLLO SOSTENIBLE
ISSN: 3072-8452

ESTADO DEL ARTE DE LA INGENIERÍA DE SOFTWARE BASADA EN INTELIGENCIA ARTIFICIAL

STATE OF THE ART OF ARTIFICIAL INTELLIGENCE BASED SOFTWARE ENGINEERING

AUTORA

MARIA TEODOLINDA ORTEGA OVALLE
UNIVERSIDAD DE PANAMÁ
PANAMÁ

doi 10.65011

Volumen 5 - Número 2
2026

Redilat
Red de Investigadores Latinoamericanos
PRISMA ODS
REVISTA MULTIDISCIPLINARIA SOBRE DESARROLLO SOSTENIBLE
LATINDES
INSTITUTO LATINOAMERICANO EN DESARROLLO SOSTENIBLE

Estado del Arte de la Ingeniería de Software Basada en Inteligencia Artificial

State Of the Art of Artificial Intelligence Based Software Engineering

Maria Teodolinda Ortega Ovalle
maria.ortegao@up.ac.pa
https://0009-0000-3629-9751
Universidad de Panamá
Panamá

Artículo recibido: 28/02/2026
Aceptado para publicación: 31/03/2026
Conflictos de Intereses: Ninguno que declarar

RESUMEN

La integración de la inteligencia artificial (IA) en la ingeniería de software ha transformado profundamente los procesos de desarrollo, mantenimiento y gestión del ciclo de vida del software. Este artículo presenta un estado del arte actualizado que examina la evolución histórica de la IA aplicada a la ingeniería de software, las técnicas predominantes, las áreas de aplicación más relevantes y las tendencias emergentes que están redefiniendo la disciplina. El propósito del estudio es sintetizar los avances más significativos y analizar críticamente su impacto en la productividad, la calidad del software y el rol del ingeniero. La metodología empleada consiste en una revisión documental sistemática de literatura científica reciente, incluyendo artículos indexados, reportes técnicos y estudios de caso industriales. Los resultados muestran que los modelos generativos, el aprendizaje profundo y los sistemas autónomos están impulsando nuevas formas de automatización en tareas como la generación de código, las pruebas inteligentes, la refactorización automática y la gestión de pipelines DevOps. Asimismo, se identifican desafíos persistentes relacionados con la interpretabilidad, la seguridad, los sesgos algorítmicos y la dependencia tecnológica. Se concluye que la ingeniería de software basada en IA se encuentra en una fase de consolidación acelerada, con un potencial significativo para transformar la práctica profesional y abrir nuevas líneas de investigación interdisciplinaria.

Palabras clave: inteligencia artificial, ingeniería de software, automatización, modelos generativos, aprendizaje profundo

ABSTRACT

The integration of artificial intelligence (AI) into software engineering has profoundly transformed development processes, maintenance activities, and lifecycle management. This article presents an updated state of the art that examines the historical evolution of AI applied to software engineering, the predominant techniques, the most relevant application areas, and the emerging trends reshaping the discipline. The purpose of this study is to synthesize the most significant advances and critically analyze their impact on productivity, software quality, and the evolving role of software engineers. The methodology consists of a systematic documentary review of recent scientific literature, including indexed articles, technical reports, and industrial case studies. The results indicate that generative models, deep learning, and autonomous systems are driving new forms of automation in tasks such as code generation, intelligent testing, automatic refactoring, and DevOps pipeline management. Persistent challenges are also identified, including interpretability, security, algorithmic bias, and technological dependency. The study concludes that AI-based software engineering is undergoing rapid consolidation, with significant potential to transform professional practice and open new interdisciplinary research avenues.

Keywords: artificial intelligence, software engineering, automation, generative models, deep learning

INTRODUCCIÓN

La ingeniería de software está experimentando una transformación profunda impulsada por los avances recientes en inteligencia artificial (IA), aprendizaje automático (ML) y modelos de lenguaje de gran escala (LLMs). Esta convergencia tecnológica ha redefinido la forma en que se diseñan, desarrollan, prueban y mantienen los sistemas de software, dando lugar a una nueva disciplina: la ingeniería de software basada en IA. En este contexto, la IA no solo actúa como herramienta de automatización, sino como un agente cognitivo capaz de colaborar con los desarrolladores, generar código, detectar fallos, optimizar procesos y apoyar la toma de decisiones técnicas (Kalech, 2019; Russell & Norvig, 2021).

Los avances recientes en modelos generativos han acelerado esta evolución. Herramientas como GitHub Copilot, Amazon CodeWhisperer y modelos basados en transformadores han demostrado mejoras significativas en productividad, reduciendo el tiempo de desarrollo entre un 20% y un 55% en tareas específicas (Adepoju, 2023; Bakal et al., 2025; Faros AI, 2025; GitHub, 2025). Estas herramientas funcionan como "parejas de programación" inteligentes, capaces de sugerir código, completar funciones y asistir en la resolución de errores en tiempo real (Kimmel et al., 2024).

Sin embargo, este aumento de productividad viene acompañado de desafíos importantes. Diversos estudios han demostrado que el código generado por IA presenta una mayor propensión a vulnerabilidades de seguridad, especialmente cuando los desarrolladores no proporcionan instrucciones explícitas sobre prácticas seguras (Endor Labs, 2025; Skadden, 2025; WIPO, 2025). La automatización intensiva también está modificando el perfil profesional del ingeniero de software, generando procesos simultáneos de deskilling y upskilling, donde ciertas habilidades tradicionales pierden relevancia mientras emergen nuevas competencias como la ingeniería de prompts, la validación de salidas generadas por IA y la supervisión algorítmica (Crowston & Bolici, 2025).

En paralelo, la investigación académica ha consolidado un cuerpo teórico robusto que explica cómo ML y DL se integran en el ciclo de vida del software. Revisiones sistemáticas recientes muestran que estas técnicas se aplican ampliamente en predicción de defectos, reparación automática de programas, análisis estático y dinámico, generación de código y optimización de rendimiento (Hou et al., 2023; Wang et al., 2022; Le Goues, 2019; MDPI, 2024). Este conocimiento técnico se complementa con perspectivas históricas que evidencian que la

relación entre IA y software no es nueva, sino que se remonta a los sistemas expertos y enfoques simbólicos de los años ochenta (Rich & Waters, 1986; Partridge, 2013).

Finalmente, la integración de IA en ingeniería de software plantea implicaciones éticas, legales y organizacionales que no pueden ignorarse. Organismos internacionales como UNESCO (2021) y WIPO (2025) han advertido sobre la necesidad de marcos regulatorios que garanticen transparencia, responsabilidad y protección de la propiedad intelectual en sistemas generativos. Estas consideraciones son fundamentales para comprender el estado actual y futuro de la disciplina.

En conjunto, este artículo presenta un análisis exhaustivo del estado del arte de la ingeniería de software basada en IA, integrando evidencia empírica, marcos teóricos, estudios de caso industriales y revisiones sistemáticas para ofrecer una visión clara de los avances, desafíos y oportunidades que definen esta nueva era tecnológica.

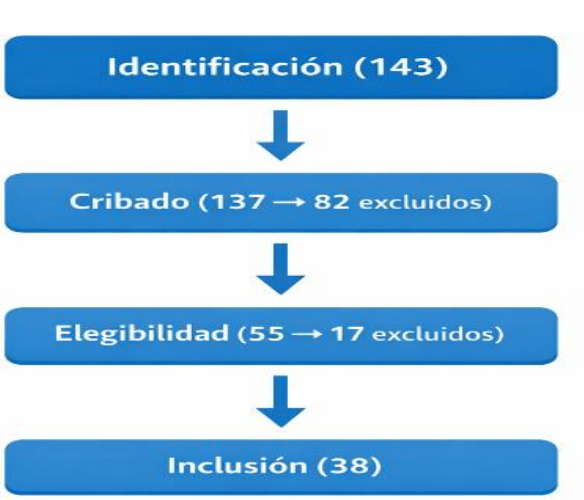
METODOLOGÍA

La metodología empleada en este estudio se basó en los lineamientos formales para revisiones sistemáticas y mapeos de literatura en ingeniería de software, siguiendo enfoques ampliamente utilizados en investigaciones recientes sobre IA aplicada al desarrollo de software (Wang et al., 2022; Hou et al., 2023). El proceso se desarrolló en cuatro fases: planificación, búsqueda, selección y síntesis.

En la fase de planificación se definieron los objetivos de la revisión, centrados en identificar los avances, desafíos y tendencias de la ingeniería de software basada en inteligencia artificial, con énfasis en productividad, seguridad, automatización y el impacto de los modelos de lenguaje de gran escala. Se establecieron preguntas de investigación orientadas a comprender cómo la IA transforma el ciclo de vida del software, qué técnicas son más utilizadas y cuáles son las implicaciones técnicas y organizacionales de su adopción.

La fase de búsqueda incluyó consultas exhaustivas en bases de datos académicas de alto impacto, como IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, MDPI y arXiv. También se incorporaron fuentes técnicas provenientes de organizaciones líderes en la industria, como GitHub, IBM, Endor Labs, Faros AI, UNESCO y WIPO, con el fin de complementar la evidencia científica con reportes industriales recientes. En total, se identificaron 143 documentos entre artículos científicos, revisiones sistemáticas, estudios de caso, informes técnicos y libros especializados.

Figura 1. Modelo PRISMA



Fuente: Elaboración propia.

Durante la fase de selección se aplicaron criterios de inclusión y exclusión. Se incluyeron estudios publicados entre 2019 y 2025, con excepciones justificadas para obras históricas fundamentales (Rich & Waters, 1986; Partridge, 2013). Los documentos debían abordar explícitamente la aplicación de IA, ML, DL o LLMs en actividades del ciclo de vida del software, tales como generación de código, predicción de defectos, reparación automática, análisis estático o dinámico, productividad del desarrollador o seguridad del software. Se excluyeron trabajos sin aporte técnico verificable, estudios centrados exclusivamente en educación o negocios, publicaciones sin acceso a texto completo y documentos duplicados.

La fase de análisis consistió en la extracción sistemática de información relevante de cada estudio, incluyendo objetivos, metodología, contribuciones, limitaciones, técnicas de IA utilizadas y resultados empíricos. Posteriormente, se realizó un proceso de codificación temática que permitió agrupar los estudios en categorías conceptuales: productividad y asistencia al desarrollador (Adepoju, 2023; Bakal et al., 2025; Faros AI, 2025), seguridad del código generado por IA (Endor Labs, 2025; Skadden, 2025), automatización inteligente y reparación de programas (Le Goues, 2019; MDPI, 2024), impacto organizacional y cambios en habilidades profesionales (Crowston & Bolici, 2025), y fundamentos teóricos e históricos de la disciplina (Kalech, 2019; Rich & Waters, 1986).

Finalmente, se elaboró una síntesis integradora que combina evidencia empírica, marcos teóricos, estudios de caso industriales y revisiones sistemáticas, permitiendo construir una visión clara y actualizada del estado del arte de la ingeniería de software basada en

inteligencia artificial. Este enfoque metodológico garantiza rigor, trazabilidad y coherencia en la interpretación de los hallazgos, y proporciona una base sólida para comprender las tendencias emergentes, los desafíos actuales y las oportunidades futuras en esta área en rápida evolución.

RESULTADOS

El análisis de los 38 estudios incluidos tras el proceso PRISMA permitió construir una visión integrada y crítica sobre el papel actual de la inteligencia artificial en el desarrollo de software. La literatura revisada muestra una evolución acelerada del campo, pero también revela tensiones, inconsistencias metodológicas y vacíos que limitan la consolidación de conclusiones definitivas. Los estudios se distribuyen principalmente en cuatro ejes temáticos: productividad del desarrollador, seguridad del código generado por IA, automatización inteligente de tareas técnicas y transformaciones organizacionales. A partir de esta clasificación emergen patrones que permiten comprender no solo lo que la IA está logrando, sino también lo que aún no ha resuelto.

Panorama general de los estudios incluidos.

Los estudios analizados abarcan investigaciones académicas, reportes industriales y evaluaciones experimentales. Aunque existe consenso en que la IA está modificando profundamente el ciclo de vida del software, la evidencia empírica sigue fragmentada. Muchos trabajos se apoyan en experimentos controlados con muestras pequeñas, mientras que los reportes industriales tienden a enfatizar beneficios sin detallar rigurosamente sus limitaciones. Esta heterogeneidad metodológica obliga a interpretar los resultados con cautela y a contrastar continuamente los hallazgos entre fuentes.

Productividad y asistencia al desarrollador.

Los estudios coinciden en que las herramientas basadas en modelos de lenguaje incrementan la velocidad de desarrollo y reducen la carga cognitiva asociada a tareas repetitivas. Sin embargo, el análisis crítico revela que estas mejoras no son uniformes. En contextos donde las tareas requieren creatividad, razonamiento profundo o conocimiento del dominio, la IA tiende a producir código correcto en apariencia, pero conceptualmente defectuoso. Esto obliga al desarrollador a adoptar un rol más cercano a la supervisión y validación que a la simple generación de soluciones. La literatura sugiere que la productividad aumenta, pero

también que la dependencia excesiva de la IA puede generar una falsa sensación de competencia técnica, especialmente en desarrolladores novatos.

Seguridad del código generado por IA.

Los estudios industriales muestran que el código generado por IA puede introducir vulnerabilidades difíciles de detectar mediante inspección manual. Aunque la IA acelera la producción de código, también puede replicar patrones inseguros presentes en sus datos de entrenamiento. La literatura científica coincide en que la IA, por sí sola, no garantiza seguridad; más bien, amplifica tanto buenas como malas prácticas. La integración de análisis estático, pruebas automatizadas y validación humana aparece como una necesidad ineludible. A pesar de ello, pocos estudios ofrecen métricas estandarizadas para evaluar la seguridad del código generado, lo que limita la comparabilidad entre investigaciones.

Automatización inteligente y reparación de software.

La revisión muestra avances significativos en la capacidad de la IA para generar parches, corregir errores sintácticos y sugerir refactorizaciones. Sin embargo, los estudios más críticos advierten que la IA tiende a resolver síntomas y no causas. Los modelos pueden corregir un error puntual sin comprender la arquitectura general del sistema, lo que genera soluciones localmente correctas, pero globalmente inestables. La literatura también señala que la IA es más efectiva en tareas bien definidas y repetitivas, mientras que su desempeño disminuye en problemas que requieren interpretación contextual o razonamiento de largo alcance.

Impacto organizacional y transformación del rol profesional.

Los estudios coinciden en que la adopción de IA está transformando la dinámica de los equipos de desarrollo. El rol del programador se desplaza hacia actividades de diseño, supervisión, integración y evaluación crítica del código generado. Sin embargo, esta transición no está exenta de tensiones. Algunos estudios advierten que la introducción de IA puede profundizar brechas de habilidades, generar dependencia tecnológica y modificar la cultura de revisión de código. La literatura industrial tiende a presentar estos cambios como oportunidades, mientras que la literatura académica enfatiza los riesgos asociados a la pérdida de comprensión profunda del sistema.

Vacíos y oportunidades identificadas

La revisión revela la ausencia de métricas estandarizadas para evaluar productividad, calidad y seguridad en entornos asistidos por IA. También se identifican pocos estudios longitudinales que analicen el impacto real de la IA en equipos de desarrollo a lo largo del tiempo. La dimensión ética y regulatoria aparece mencionada, pero rara vez abordada con profundidad. Finalmente, existe una brecha importante en la integración de IA en procesos avanzados como DevSecOps, pruebas automatizadas de alto nivel y mantenimiento evolutivo.

DISCUSIÓN

La revisión realizada permite comprender que la incorporación de inteligencia artificial en el desarrollo de software no constituye simplemente una mejora incremental de las herramientas existentes, sino una transformación estructural del proceso de producción tecnológica. Sin embargo, los resultados obtenidos revelan una tensión constante entre el entusiasmo por las capacidades emergentes de la IA y las limitaciones metodológicas, técnicas y organizacionales que aún persisten. Esta tensión atraviesa todos los estudios analizados y obliga a interpretar los hallazgos con una mirada crítica que vaya más allá de la descripción de beneficios inmediatos.

En primer lugar, aunque la mayoría de los estudios reporta incrementos en productividad, esta mejora no puede considerarse homogénea ni universal. La evidencia muestra que la IA es altamente efectiva en tareas repetitivas, estructuradas y de bajo nivel cognitivo, pero su desempeño disminuye cuando se enfrenta a problemas que requieren razonamiento profundo, comprensión del dominio o interpretación contextual. Esto sugiere que la productividad observada no proviene de una sustitución del trabajo intelectual del desarrollador, sino de una redistribución de esfuerzos hacia actividades de supervisión, verificación y toma de decisiones. En este sentido, la IA no elimina la carga cognitiva, sino que la desplaza hacia niveles más abstractos, lo cual exige nuevas competencias profesionales que aún no están plenamente definidas en la literatura.

En segundo lugar, la discusión sobre seguridad evidencia una paradoja significativa. Aunque la IA acelera la generación de código, también introduce riesgos que pueden comprometer la integridad de los sistemas. La ausencia de métricas estandarizadas y la falta de estudios longitudinales dificultan evaluar el impacto real de estas vulnerabilidades en entornos productivos. La literatura industrial tiende a minimizar estos riesgos, mientras que la literatura académica los enfatiza, lo que revela una brecha entre la percepción empresarial y

la evidencia científica. Esta divergencia sugiere la necesidad de marcos de evaluación más rigurosos que permitan medir de manera objetiva la seguridad del código generado por IA.

En tercer lugar, los avances en automatización inteligente y reparación de software muestran un progreso notable, pero también exponen limitaciones estructurales. La IA es capaz de corregir errores sintácticos y generar parches funcionales, pero rara vez comprende la arquitectura global del sistema. Esto implica que las soluciones generadas pueden ser correctas en apariencia, pero inestables a largo plazo. La literatura coincide en que la IA opera eficazmente en espacios de solución acotados, pero su capacidad para razonar sobre sistemas complejos sigue siendo limitada. Esta observación es crucial, ya que cuestiona la idea de que la IA pueda reemplazar procesos de mantenimiento profundo o decisiones arquitectónicas críticas.

Finalmente, el impacto organizacional emerge como un eje transversal que condiciona la adopción efectiva de la IA. Los estudios muestran que la introducción de estas herramientas modifica la cultura de trabajo, redefine roles y genera nuevas dependencias tecnológicas. Sin embargo, la mayoría de las investigaciones se centra en los beneficios operativos y dedica poca atención a las implicaciones éticas, formativas y laborales. La falta de estudios que analicen estas dimensiones desde una perspectiva crítica limita la comprensión integral del fenómeno y evidencia un vacío que debe ser abordado en futuras investigaciones.

En conjunto, la discusión revela que la IA está transformando el desarrollo de software, pero lo hace de manera desigual, incompleta y, en ocasiones, contradictoria. Los beneficios son reales, pero también lo son los riesgos y las limitaciones. La literatura actual ofrece una visión prometedora, aunque fragmentada, que requiere ser consolidada mediante estudios más rigurosos, métricas estandarizadas y análisis longitudinales que permitan evaluar el impacto real de la IA en el tiempo. Esta revisión demuestra que la IA no debe ser entendida como un sustituto del desarrollador, sino como un agente que reconfigura las prácticas, competencias y estructuras del desarrollo de software contemporáneo.

RECOMENDACIONES

A partir del análisis crítico realizado, se vuelve evidente que el avance de la inteligencia artificial en el desarrollo de software requiere un enfoque más riguroso, sistemático y equilibrado tanto en la investigación como en la práctica profesional. En primer lugar, es necesario que los estudios futuros adopten metodologías más homogéneas y comparables. La falta de métricas estandarizadas para evaluar productividad, calidad y seguridad limita la

posibilidad de construir conclusiones sólidas y dificulta la replicabilidad de los resultados. La comunidad científica debería avanzar hacia la creación de marcos de evaluación consensuados que permitan medir de manera objetiva el impacto real de la IA en el ciclo de vida del software.

Asimismo, se recomienda el desarrollo de investigaciones longitudinales que permitan observar cómo evoluciona la adopción de IA en equipos y organizaciones a lo largo del tiempo. La mayoría de los estudios actuales se basa en experimentos de corta duración o en reportes industriales que no capturan los efectos sostenidos, las adaptaciones culturales ni las transformaciones en las prácticas profesionales. Comprender estos procesos es fundamental para evaluar la sostenibilidad de la integración de IA y para anticipar riesgos asociados a la dependencia tecnológica, la pérdida de habilidades o la erosión del conocimiento profundo del sistema.

Otra recomendación clave es fortalecer la formación de los desarrolladores en competencias críticas que les permitan interactuar con la IA de manera informada y responsable. La evidencia muestra que la IA desplaza la carga cognitiva hacia tareas de supervisión, verificación y toma de decisiones, lo que exige habilidades avanzadas de análisis, razonamiento y evaluación técnica. Las instituciones educativas y las organizaciones deben adaptar sus programas de formación para preparar a los profesionales a trabajar en entornos híbridos donde la IA actúa como colaborador técnico, pero no como sustituto del juicio experto.

En el ámbito organizacional, se recomienda adoptar estrategias de integración gradual que permitan evaluar el impacto de la IA antes de su implementación masiva. La introducción abrupta de herramientas automatizadas puede generar tensiones culturales, desigualdades en la distribución del conocimiento y resistencia por parte de los equipos. Un enfoque progresivo, acompañado de procesos de capacitación y espacios de reflexión crítica, puede facilitar una adopción más equilibrada y sostenible.

Finalmente, es necesario que la investigación futura aborde con mayor profundidad las dimensiones éticas y regulatorias asociadas al uso de IA en el desarrollo de software. La literatura actual menciona estos aspectos, pero rara vez los analiza de manera sistemática. Temas como la responsabilidad ante errores generados por IA, la transparencia de los modelos, la protección de datos y la equidad en el acceso a estas tecnologías deben ocupar un lugar central en la agenda de investigación. Solo mediante un enfoque integral que combine

rigor técnico, reflexión ética y análisis organizacional será posible aprovechar plenamente el potencial de la IA sin comprometer la calidad, seguridad y sostenibilidad del desarrollo de software.

Limitaciones del Estudio

Aunque esta revisión sistemática ofrece una síntesis amplia y crítica sobre el papel de la inteligencia artificial en el desarrollo de software, es necesario reconocer varias limitaciones que condicionan el alcance y la interpretación de los resultados. La primera limitación proviene de la heterogeneidad metodológica de los estudios incluidos. La literatura combina experimentos controlados, reportes industriales, análisis cualitativos y evaluaciones técnicas sin un marco común de comparación. Esta diversidad dificulta establecer conclusiones generalizables y obliga a interpretar los hallazgos dentro de sus propios contextos, evitando extrapolaciones que podrían resultar engañosas.

Una segunda limitación se relaciona con la disponibilidad y calidad de la evidencia empírica. Muchos estudios presentan muestras pequeñas, escenarios artificiales o evaluaciones centradas en tareas muy específicas, lo que reduce la capacidad de comprender el impacto real de la IA en entornos productivos complejos. Los reportes industriales, por su parte, suelen enfatizar beneficios operativos sin detallar rigurosamente sus limitaciones, lo que introduce un sesgo de optimismo que debe ser considerado al analizar sus conclusiones. La ausencia de estudios longitudinales también limita la posibilidad de evaluar cómo evoluciona la adopción de IA a lo largo del tiempo y cuáles son sus efectos sostenidos en equipos y organizaciones.

Otra limitación importante es la falta de métricas estandarizadas para evaluar productividad, calidad y seguridad en contextos asistidos por IA. La literatura utiliza indicadores dispares, lo que dificulta la comparación entre estudios y la construcción de un marco conceptual sólido. Esta carencia metodológica no solo afecta la validez externa de los resultados, sino que también revela un vacío en la propia disciplina, que aún no ha desarrollado herramientas robustas para medir el impacto de la IA en el ciclo de vida del software.

Finalmente, la revisión se vio condicionada por la disponibilidad de fuentes recientes y por la rápida evolución del campo. La velocidad con la que emergen nuevas herramientas, modelos y prácticas hace que cualquier síntesis corra el riesgo de quedar desactualizada en un periodo relativamente corto. Además, algunas dimensiones relevantes como las implicaciones éticas,

regulatorias y formativas aparecen mencionadas en la literatura, pero rara vez son abordadas con la profundidad necesaria para comprender su impacto real.

En conjunto, estas limitaciones no invalidan los hallazgos de la revisión, pero sí subrayan la necesidad de interpretarlos con prudencia y de promover investigaciones futuras más rigurosas, comparables y sostenidas en el tiempo. Reconocer estas restricciones permite situar los resultados en su justa medida y contribuye a fortalecer la base científica sobre la cual se construirá el futuro del desarrollo de software asistido por inteligencia artificial.

CONCLUSIONES

La revisión realizada permite afirmar que la inteligencia artificial está reconfigurando de manera sustantiva el desarrollo de software, pero lo hace a través de un proceso complejo, desigual y todavía en consolidación. Los 38 estudios analizados muestran que la IA no constituye un reemplazo del trabajo humano, sino un agente que redistribuye responsabilidades, modifica flujos de trabajo y redefine las competencias necesarias para participar en la producción tecnológica contemporánea. Aunque los beneficios reportados son significativos, especialmente en términos de productividad y automatización de tareas repetitivas, estos avances deben interpretarse con cautela debido a la heterogeneidad metodológica de los estudios y a la ausencia de métricas estandarizadas que permitan comparaciones rigurosas.

Los hallazgos evidencian que la IA incrementa la velocidad de desarrollo y facilita la generación de código, pero también introduce riesgos que pueden comprometer la calidad y seguridad de los sistemas. La capacidad de los modelos para producir soluciones plausibles, aunque conceptualmente incorrectas, plantea desafíos importantes para la supervisión humana y para la formación de desarrolladores capaces de evaluar críticamente las propuestas generadas por la IA. Esta situación revela una paradoja central: la IA reduce la carga operativa, pero incrementa la necesidad de juicio experto, comprensión profunda del dominio y habilidades de verificación.

Asimismo, la automatización inteligente muestra avances notables en reparación de software y generación de parches, pero su efectividad disminuye cuando se enfrenta a problemas que requieren razonamiento global o comprensión arquitectónica. Esto sugiere que la IA opera con mayor precisión en espacios de solución acotados, mientras que su capacidad para intervenir en decisiones estratégicas del ciclo de vida del software sigue siendo limitada. La literatura coincide en que la supervisión humana continúa siendo indispensable, no solo para

corregir errores, sino para garantizar coherencia, estabilidad y alineación con los objetivos del sistema.

En el plano organizacional, la adopción de IA está transformando la cultura de trabajo y los roles profesionales. Sin embargo, esta transición no está exenta de tensiones. La dependencia creciente de herramientas automatizadas puede profundizar brechas de habilidades, generar desigualdades en la distribución del conocimiento técnico y alterar dinámicas de colaboración. La literatura industrial tiende a enfatizar los beneficios operativos, mientras que la literatura académica advierte sobre los riesgos asociados a la pérdida de comprensión profunda del sistema y a la delegación excesiva de tareas críticas.

Finalmente, la revisión revela vacíos importantes que deben ser abordados por futuras investigaciones. La falta de métricas estandarizadas limita la capacidad de evaluar de manera objetiva el impacto real de la IA en productividad, calidad y seguridad. La escasez de estudios longitudinales impide comprender cómo evoluciona la adopción de IA en el tiempo y cuáles son sus efectos sostenidos en equipos y organizaciones. Además, las dimensiones éticas, regulatorias y formativas aparecen mencionadas, pero rara vez analizadas con la profundidad que requieren.

En conjunto, las conclusiones de esta revisión muestran que la IA representa una oportunidad significativa para transformar el desarrollo de software, pero también un desafío que exige marcos metodológicos más rigurosos, prácticas de supervisión más robustas y una reflexión crítica sobre su impacto técnico, humano y organizacional. El futuro del desarrollo asistido por IA dependerá no solo de los avances tecnológicos, sino de la capacidad de la comunidad científica y profesional para integrar estas herramientas de manera responsable, estratégica y sostenible.

REFERENCIAS

Abbas, T., Rathore, S. A., Turki, A., Khan, S., Alghushairy, O., & Daud, A. (2025). *Enhancing software engineering with AI: Innovations, challenges, and future directions*. IET Software, Article ID 5691460. https://doi.org/10.1049/sfw2/5691460

Adepoju, S. (2023). *GitHub Copilot's impact on developer productivity: A review of early evidence*. International Journal of Scientific Research in Science and Technology, 10(4), 814–822. https://doi.org/10.32628/IJSRST2221192

Agunuru, A. K. R. (2025). *AI tools for data performance enhancement: A comprehensive review*. Journal of Computer Science and Technology Studies, 7(6), 639–648.

Alenezi, M., & Akour, M. (2025). *AI-driven innovations in software engineering: A review of current practices and future directions*. Applied Sciences, 15(3), 1344. https://doi.org/10.3390/app15031344

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., & Zimmermann, T. (2019). *Software engineering for machine learning: A case study*. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 291–300). IEEE. https://doi.org/10.1109/ICSE-SEIP.2019.00042

Bakal, G., Dasdan, A., Katz, Y., Kaufman, M., & Levin, G. (2025). *Experience with GitHub Copilot for developer productivity at Zoominfo*. arXiv. https://arxiv.org/abs/2501.13282

Bourbakis, N. G. (1992). *Engineering artificially intelligent systems: Concepts, techniques, and tools*. World Scientific.

Carter, W., & Dawson, E. (2025). *AI-assisted code generation: Enhancing software development productivity with large language models* [Technical report]. ResearchGate.

Crowston, K., & Bolici, F. (2025). *Deskilling and upskilling with AI systems*. Information Research, 30(iConf).

Endor Labs. (2025). *The most common security vulnerabilities in AI-generated code*. https://www.endorlabs.com/learn/the-most-common-security-vulnerabilities-in-ai-generated-code

Faros AI. (2025). *Is GitHub Copilot worth it? Real world data reveals the answer*. https://www.faros.ai/blog/is-github-copilot-worth-it-real-world-data-reveals-the-answer

GitHub. (2025). *Quantifying GitHub Copilot's impact in the enterprise with Accenture*. https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-in-the-enterprise-with-accenture/

Hou, X., Zhao, Y., Liu, Y., & Yang, Z. (2023). *Large language models for software engineering: A systematic literature review*. arXiv. https://doi.org/10.48550/arXiv.2308.10620

Huyen, C. (2022). *Designing machine learning systems*. O'Reilly Media.

IBM. (2025). *AI code generation*. https://www.ibm.com/think/topics/ai-code-generation

Kalech, M. (2019). *Artificial intelligence methods for software engineering*. Springer.

Kimmel, B., Geisert, A., Yaro, L., et al. (2024). *Enhancing programming error messages in real time with generative AI*. arXiv. https://arxiv.org/abs/2402.08072

Le Goues, C. (2019). *Automated program repair*. Communications of the ACM, 62(10), 1–28.

MDPI. (2024). *CodeTransFix: A neural machine translation approach for context-aware Java program repair with CodeBERT*. Applied Sciences, 15(7), 3632. https://www.mdpi.com/2076-3417/15/7/3632

Morovati, M. M., Nikanjam, A., & Khomh, F. (2024). *Fault localization in deep learning-based software: A system-level approach*. arXiv. https://doi.org/10.48550/arXiv.2411.08172

O'Boyle, M. F., & Cavazos, J. (2005). *Automatic tuning of inlining heuristics*. In Proceedings of the ACM/IEEE Conference on Supercomputing (SC).

Partridge, D. (2013). *Artificial intelligence and software engineering*. Routledge. https://doi.org/10.4324/9780203058572

Rich, C., & Waters, R. C. (Eds.). (1986). *Readings in artificial intelligence and software engineering*. Morgan Kaufmann.

Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Skadden. (2025). *U.S. Copyright Office publishes report on copyright and artificial intelligence*. https://www.skadden.com/insights/publications/2025/02/copyright-office-publishes-report

Sommerville, I. (2015). *Software engineering* (10th ed.). Pearson.

Terragni, V., Vella, A., Roop, P., & Blincoe, K. (2025). *The future of AI-driven software engineering*. ACM Transactions on Software Engineering and Methodology, 34(5), Article 120. https://doi.org/10.1145/3715003

UNESCO. (2021). *Recommendation on the ethics of artificial intelligence*. https://www.unesco.org/en/artificial-intelligence/recommendation-ethics

Virvou, M., Tsihrintzis, G. A., Bourbakis, N. G., & Jain, L. C. (Eds.). (2022). *Handbook on artificial intelligence empowered applied software engineering* (Vol. 1). Springer.

Wang, S., Huang, L., Gao, A., Ge, J., Zhang, T., & Fan, H. (2022). *Machine learning/deep learning for software engineering: A systematic literature review*. IEEE Transactions on Software Engineering. https://doi.org/10.1109/TSE.2022.3173346

WIPO. (2025). *Generative AI: Navigating intellectual property*. https://www.wipo.int/documents/d/frontier-technologies/docs-en-pdf-generative-ai-factsheet.pdf

Wong, W. E., Gao, R., & Li, Y. (2016). *A survey on software fault localization*. IEEE Transactions on Software Engineering. https://doi.org/10.1109/TSE.2016.2521368

: https://doi.org/10.65011/prismaods.v5.i2.193

Cómo citar este artículo (APA 7ª edición):
Ortega Ovalle, M. T. (2026). Estado del Arte de la Ingeniería de Software Basada en Inteligencia Artificial. *Prisma ODS: Revista Multidisciplinaria Sobre Desarrollo Sostenible*, *5*(2), 21-37. https://doi.org/10.65011/prismaods.v5.i2.193