Revista Científica Multidisciplinar G-ner@ndo

ISSN: 2806-5905

Herramientas de inteligencia artificial usadas en el desarrollo de software. una revisión sistemática de la literatura.

Artifice intelligence tools used in software development. a systematic revision of literature.

Alfonso Ramon García Alcívar, Eddy Alejandro Loor Navia

CIENCIA E INNOVACIÓN EN DIVERSAS DISCIPLINAS CIENTÍFICAS.

Enero - Junio, V°6-N°1; 2025

Recibido: 08/04/2025
Aceptado: 09/04/2025
Publicado: 30/06/2025

PAIS

- Ecuador – Portoviejo
- Ecuador – Portoviejo

INSTITUCION

- Universidad Técnica de Manabí
- Universidad Técnica de Manabí

CORREO:

✉ alfongarc_15@hotmail.com
✉ alejandro.loor@utm.edu.ec

ORCID:

https://orcid.org/0009-0004-3525-8922
https://orcid.org/0000-0001-6670-835X

FORMATO DE CITA APA.

Garcia, A. & Loor, E. (2025). Herramientas de inteligencia artificial usadas en el desarrollo de software. una revisión sistemática de la literatura. *Revista G-ner@ndo, V°6 (N°1,). 4035 –4060.*

Resumen

La inteligencia artificial (IA) está revolucionando el desarrollo de software, ofreciendo soluciones que mejoran la eficiencia, calidad y seguridad en diversas etapas del ciclo de vida del software. Este artículo examina las herramientas y enfoques de IA más relevantes, como los asistentes de codificación, modelos de deep learning para detección de errores, y sistemas para optimización de pruebas y gestión de requisitos. A través de una revisión sistemática de la literatura, se identifican patrones y tendencias, destacando los beneficios de la automatización en la programación, la predicción de defectos y la mejora en la productividad de los equipos de desarrollo. Este trabajo proporciona una visión integral sobre el impacto de la IA en la ingeniería de software.

Palabras clave: Inteligencia Artificial, Desarrollo de Software, Automatización, Detección de Errores, Optimización de Pruebas.

Abstract

Artificial intelligence (AI) is revolutionizing software development, offering solutions that improve efficiency, quality, and security at various stages of the software lifecycle. This article examines the most relevant AI tools and approaches, such as coding assistants, deep learning models for error detection, and systems for test optimization and requirements management. Through a systematic literature review, patterns and trends are identified, highlighting the benefits of automation in programming, defect prediction, and improved productivity for development teams.This paper provides a comprehensive overview of the impact of AI on software engineering.

Keywords: Artificial Intelligence, Software Development, Automation, Error Detection, Test Optimization.

Introducción

En los últimos años, la inteligencia artificial (IA) ha transformado diversas industrias, y el desarrollo de software no ha sido la excepción (Kuhail et al., 2024). La creciente complejidad de los proyectos y la necesidad de soluciones más ágiles han impulsado la adopción de tecnologías de IA en todas las etapas del ciclo de vida del desarrollo. Herramientas como ChatGPT y OpenAI Codex están automatizando tareas repetitivas y permitiendo a los desarrolladores enfocarse en problemas estratégicos, lo que se traduce en un aumento significativo de la productividad (Bamohabbat Chafjiri et al., 2024). Además, enfoques basados en deep learning están mejorando la detección de errores y vulnerabilidades, elevando la seguridad y calidad del software (Asikainen & Männistö, 2022). Sin embargo, a pesar de estos avances, la integración de la IA en el desarrollo de software presenta desafíos importantes, como la interpretabilidad de los modelos y la mitigación de sesgos, aspectos fundamentales para su adopción responsable (Sivakumar et al., 2024). Este artículo analiza cómo estas herramientas están revolucionando el desarrollo de software, destacando tanto sus beneficios como los retos que deben abordarse para maximizar su impacto.

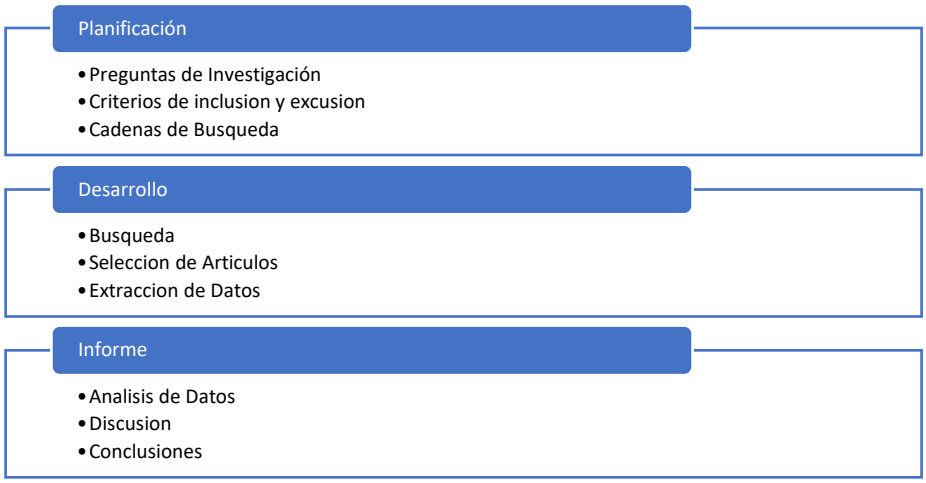
Métodos y Materiales

El presente articulo emplea un enfoque metodológico mixto, combinando estrategias inductivas y deductivas. En la fase deductiva se establecen criterios de inclusión y exclusión basados en hipótesis y marcos teóricos, lo que permite filtrar la literatura científica relevante. Posteriormente, mediante un enfoque inductivo, se extraen e identifican patrones, temas y tendencias emergentes en las herramientas de IA aplicadas al desarrollo de software. Este proceso dual garantiza una rigurosa selección y análisis de la información.

La presente investigación se centra en una revisión sistemática de la literatura, siguiendo el flujo PRISMA que abarca la identificación, selección, elegibilidad e inclusión de estudios,

facilitando una síntesis coherente y estructurada de los hallazgos. La revisión estará dividida en tres puntos: Planificación, Desarrollo e Informe (Figura 1).

Figura 1: *Esquema del proceso.*



Elaborado: Autor

Planificación.

En esta etapa se definieron las preguntas de investigación, los criterios de inclusión y exclusión para seleccionar los estudios relevantes.

Preguntas de Investigación

Para enmarcar la investigación relacionada con la problemática planteada, se formularon las siguientes preguntas de investigación (PI):

PI1: ¿Cuáles son las herramientas de inteligencia artificial documentadas en la literatura científica y técnica para su aplicación en el desarrollo de software?

PI2 ¿Qué características individuales, capacidades específicas y áreas de aplicación se destacan para cada herramienta de IA utilizada en el desarrollo de software según la literatura revisada?

PI3 ¿Cómo se comparan las herramientas de inteligencia artificial en términos de eficiencia, adaptabilidad y efectividad en diferentes entornos y etapas del ciclo de vida del software?

PI4 ¿Qué marco comparativo puede sintetizarse de los hallazgos en la literatura que sirva de referencia para profesionales del desarrollo de software y entidades interesadas?

Criterios de Inclusión y Exclusión

Los criterios de inclusión y exclusión se muestran en la Tabla 1, donde, se buscarán estudios publicados en los últimos 5 años. Investigaciones que detallen el uso de herramientas de IA específicas en el ciclo de vida del desarrollo de software. Artículos que presenten casos de estudio o experimentos prácticos con resultados medibles sobre el impacto de las herramientas de IA en el desarrollo de software. Investigaciones que aborden diferentes enfoques de IA, como aprendizaje automático, procesamiento del lenguaje natural o visión por computadora, aplicados al desarrollo de software. Estudios que proporcionen datos cuantitativos o cualitativos sobre la eficacia, eficiencia o mejoras gracias al uso de herramientas de IA.

Los criterios de exclusión se centran en artículos que no estén relacionados directamente con el desarrollo de software. Estudios que no presenten información detallada sobre las herramientas de IA utilizadas. Investigaciones sin acceso al texto completo o que carezcan de rigurosidad metodológica. Publicaciones que no estén disponibles en idiomas accesibles para el análisis (por ejemplo, idiomas sin traducción o acceso limitado). Estudios obsoletos que no reflejen las tendencias actuales en herramientas de IA para el desarrollo de software.

Se realizó la identificación y selección de estudios mediante búsquedas exhaustivas en bibliotecas digitales tales como IEEE Xplore, Redalyc y ScienceDirect. Se eligieron estas fuentes por su capacidad para abarcar un amplio espectro de publicaciones en el ámbito de las ciencias de la informática, garantizando así una base de datos sólida y consistente. A continuación, en la

Tabla 1 se muestran las cadenas de búsqueda y hallazgos que se encontraron a partir del año 2019 lo que garantiza una información actualizada.

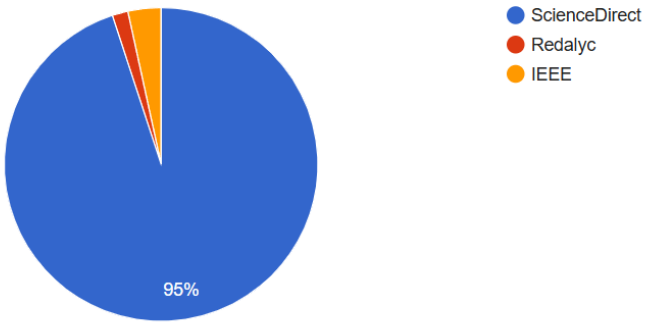
Tabla 1: *Cadenas de búsqueda y resultados obtenidos.*

Cadena de Busqueda	IEEE Xplore	ScienceDirect	Redalyc
("artificial intelligence" OR "AI") AND ("software development lifecycle" OR "software engineering process") AND ("efficiency" OR "effectiveness" OR "adaptability")	4	135	4
("artificial intelligence" OR "AI") AND ("software development lifecycle" OR "SDLC" OR "software engineering process") AND ("automation" OR "productivity" OR "performance")	8	226	4
("artificial intelligence" OR "AI") AND ("software development lifecycle" OR "SDLC" OR "software engineering process") AND ("quality assurance" OR "testing" OR "continuous integration")	9	223	2
TOTAL	21	584	10

Elaborado: Autor

Una vez aplicadas las cadenas de búsqueda en las bibliotecas Digitales se encontraron un total de 615 artículos.

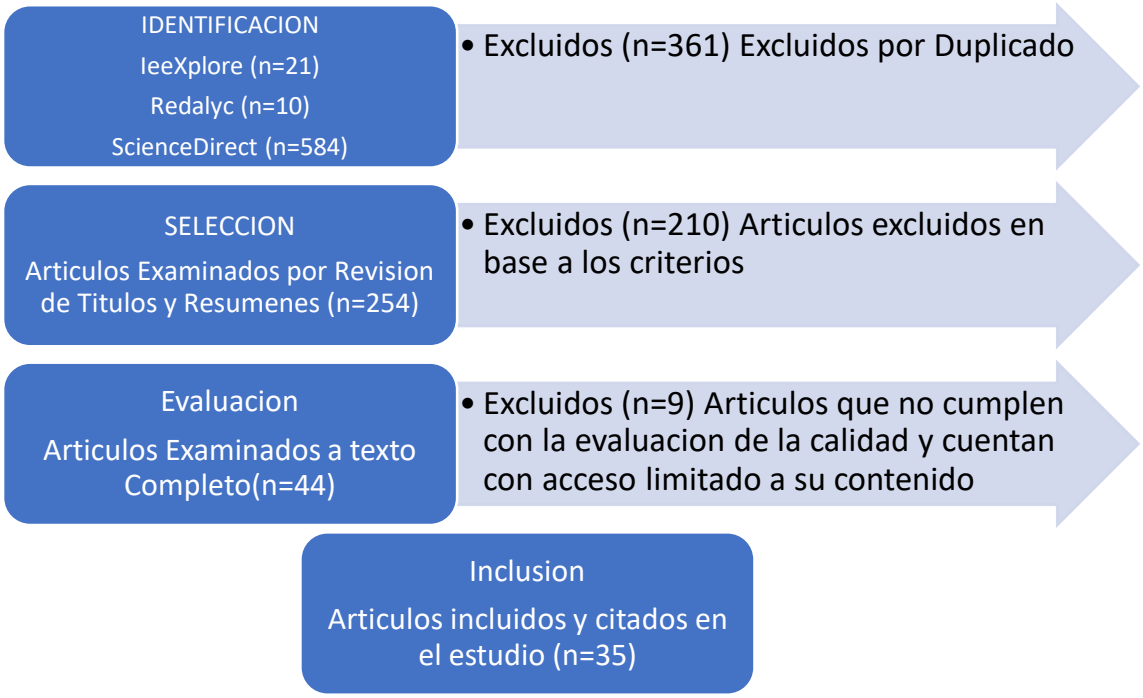
Figura 2: *Artículos por Fuente.*



Elaborado: Autor

Después de obtener estos resultados, se importaron los datos de cada búsqueda en formato BibTeX para ser procesados con Parcifal y así poder aplicar los criterios de inclusión y exclusión. A continuación, en la Figura 3 se Detalla el flujograma Prisma aplicado en la revisión.

Figura 3: *Flujograma Prisma.*



Elaborado*:* Autor

Al aplicar la selección que tuvo como base los criterios de inclusión y exclusión, se seleccionaron un total de 35 artículos, los cuales cumplen dichos criterios. Estos artículos serán los que fundamenten el SLR.

El proceso de selección se llevó a cabo a partir de un total de 615 artículos recopilados de tres bases de datos:

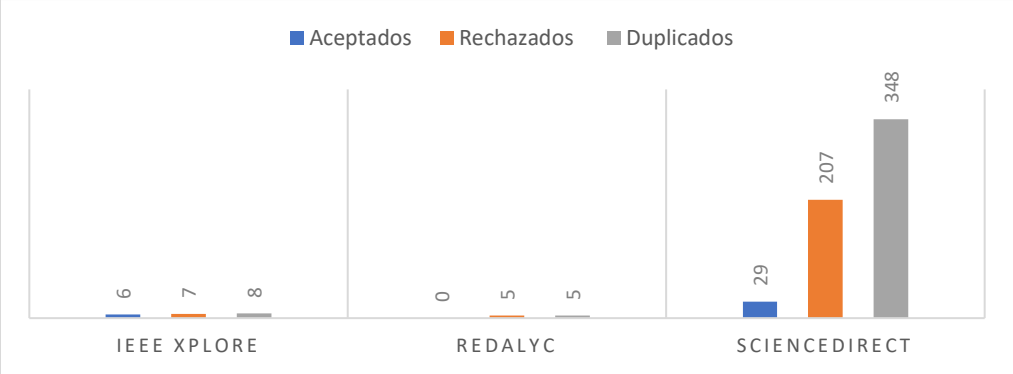
IEEE Xplore: Se identificaron 21 artículos, de los cuales 6 fueron aceptados, 7 fueron rechazados y 8 fueron descartados por ser duplicados.

Redalyc: Se encontraron 10 artículos, de los cuales ninguno fue aceptado, 5 fueron rechazados y 5 fueron eliminados por ser duplicados.

ScienceDirect: Se extrajeron 584 artículos, de los cuales 29 fueron aceptados, 207 fueron rechazados y 348 fueron eliminados por ser duplicados.

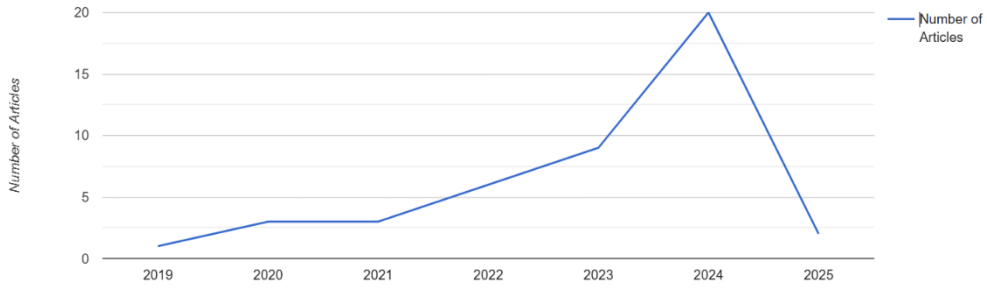
Este proceso permitió depurar la selección inicial, asegurando que los artículos incluidos cumplan con los criterios de calidad y relevancia para el estudio.

Figura 4: *Artículos duplicados, rechazados y aceptados por Fuentes*



Elaborado: Autor

Figura 5: *Artículos finales por Año*



Elaborado: Autor

Análisis de Resultados

Se extrajo y sintetizó la información clave de los documentos aceptados o incluidos, lo que permitió comprender el estado actual del uso de herramientas de inteligencia artificial en el desarrollo de software. Esto facilitó el análisis de las principales aplicaciones, enfoques y beneficios de estas herramientas en las distintas etapas del ciclo de vida del desarrollo de software. La revisión sistemática identificó una diversidad de herramientas y enfoques de inteligencia artificial (IA), entre los que se destacan soluciones avanzadas que abarcan distintas

áreas del desarrollo de software. Estas herramientas no solo mejoran la productividad y calidad en varias etapas del ciclo de vida del software, sino que también optimizan procesos clave como la codificación, la gestión de requisitos, la detección de vulnerabilidades, y la predicción de defectos:

Uno de los enfoques más notables son los modelos de lenguaje y asistentes de codificación, como ChatGPT, OpenAI Codex, DeepMind AlphaCode, y Amazon CodeWhisperer. Estas herramientas han demostrado ser eficaces para automatizar tareas repetitivas y complejas de programación, permitiendo a los desarrolladores centrarse en problemas más estratégicos. Se ha reportado que su uso puede aumentar la productividad en tareas de codificación hasta en un 55%, según diversos estudios (Kuhail et al., 2024).

En el ámbito de la detección de vulnerabilidades, se han identificado técnicas avanzadas que incorporan aprendizaje automático tradicional, profundo y por refuerzo, las cuales se aplican en el proceso de fuzzing (técnica de pruebas de software). Estas técnicas permiten una identificación más precisa de errores y fallos de seguridad en el software. Los enfoques basados en IA han demostrado ser significativamente más efectivos que los métodos tradicionales, con una mejora en la tasa de detección de vulnerabilidades que supera el 30%. (Bamohabbat Chafjiri et al., 2024).

Por otro lado, los frameworks para ingeniería de software basada en datos, como el Marco Undulate, utilizan principios de soft computing para apoyar la toma de decisiones durante el proceso de desarrollo. Estos enfoques no solo permiten una evaluación más rápida de las alternativas disponibles, sino que también han demostrado reducir el tiempo de análisis en un 40%, (Asikainen & Männistö, 2022), lo que representa un avance significativo en términos de eficiencia operativa. Además, se han documentado herramientas que optimizan el proceso de pruebas de software y la predicción de calidad, mediante algoritmos como Random Forest, SVM, y ANN. Estas técnicas permiten mejorar la eficiencia de las pruebas y predicciones, logrando un

incremento en la eficacia de las pruebas de hasta un 25%. ((Mehmood et al., 2024), (Alaswad & Poovammal, 2022)), esto es particularmente útil en fases tempranas del ciclo de vida del software, cuando la identificación de errores es crucial para asegurar un producto de alta calidad.

En términos de gestión de requisitos y errores, herramientas como RE4HCAI y modelos basados en ACO (ant colony optimization) se aplican para mejorar la asignación de tareas y la gestión de incidencias, incrementando la precisión en un 20% en la asignación de programadores a tareas específicas. ((Ahmad et al., 2023), (Kukkar et al., 2023)).

Destacan herramientas avanzadas para la generación automática de documentación y la creación de casos de seguridad, como el modelo GPT-4. Este modelo, al ser capaz de generar documentación detallada en base a las especificaciones del software, ha mostrado una reducción en el tiempo dedicado a esta tarea de hasta un 60% (Sivakumar et al., 2024), lo cual puede acelerar significativamente los ciclos de desarrollo. Adicionalmente, se identificaron herramientas especializadas en varias áreas avanzadas de desarrollo de software:

Tabla 2: Herramientas especializadas.

Evaluación de proyectos y operacionalización en la nube		
Herramientas como Orfeon y modelos basados en Deep Extreme Learning Machines	Mejoras en la precisión de las predicciones de riesgos de hasta un 35%	Esencial para la planificación y mitigación de riesgos en entornos de desarrollo en la nube.
Fuentes: (Ikram et al., 2022), (Díaz-de-Arcaya et al., 2023)		
Predicción de defectos y análisis de requisitos		
Modelos como CodeBERT, enfoques de deep learning, y soluciones híbridas	Mejoras en la detección de errores y extracción de diagramas UML con una precisión de hasta 92%	Mejora significativamente la calidad del software.
Fuentes: (Ghafoor Hussain et al., 2025), (Mazhar et al., 2024), (Babaalla et al., 2024)		

Optimización del esfuerzo de pruebas		
Algoritmos como Gradient Boosting optimizados mediante Differential Evolution	Reducción de la sobreestimación del esfuerzo en las pruebas en un 18%	Optimiza los recursos y el tiempo de pruebas.
Fuentes: (Sánchez-García et al., 2023)		

Elaborado: Autor

Además, se identificaron nuevas herramientas que abarcan varias áreas de innovación en el desarrollo de software:

Tabla 3: *Nuevas herramientas*

Modelos híbridos para automatización en DevOps	Lograron mejoras del 40% en la eficiencia de implementación de pipelines de CI/CD.	Optimizando la calidad del software a lo largo de su ciclo de vida.
Fuentes: (Senanayake et al., 2024), (Alizadehsani et al., 2023)		
Integración de IA en sistemas de gestión de configuración	Mostraron una reducción del 30% en errores de control de versiones	Optimizando la calidad del software a lo largo de su ciclo de vida.
Fuentes: (Golshanrad & Faghih, 2024)		
Algoritmos para la mejora en revisión de código automatizada	Lograron una reducción del 35% en el tiempo de revisión del código	Mejorando la eficiencia del equipo de desarrollo.
Fuentes: (Martins et al., 2023), (Draz et al., 2021)		
Análisis predictivo para estimación de esfuerzo en el SDLC **Fuentes:** (Nazir et al., 2024)	Alcanzaron una precisión del 85% en las estimaciones de esfuerzo	Contribuyendo a una mejor planificación de proyectos.
Herramientas de IA para la identificación y prevención de defectos en código fuente	Permitieron reducir los defectos en producción en un 25%	Mejorando la calidad final del software.
Fuentes: (Pauzi & Capiluppi, 2023), (Eramo et al., 2024)		
Enfoques basados en redes neuronales para refactorización de código	Lograron una mejora del 40% en la legibilidad y rendimiento del código	Facilitando el mantenimiento y la evolución del software.
Fuentes: (Tahvili et al., 2020), (Dong et al., 2025)		

Machine learning para mejora de la calidad del software en proyectos colaborativos	Optimizando la asignación de tareas en equipos distribuidos en un 30%	Mejorando la eficiencia del trabajo en equipos ágiles y distribuidos geográficamente

Fuentes: (Tameswar et al., 2022)

Elaborado: Autor

La diversidad de herramientas identificadas en la revisión sistemática refleja el impacto transformador de la inteligencia artificial en la industria del software. Estas herramientas abarcan desde la mejora en la productividad de los desarrolladores hasta la optimización de la calidad del software y la reducción de costos operativos, facilitando una integración más fluida de la IA en los procesos de desarrollo de software.

En la revisión sistemática, También se identificaron diversas herramientas de inteligencia artificial que presentan características individuales, capacidades específicas y áreas de aplicación bien definidas dentro del desarrollo de software. Estas herramientas han demostrado ser clave en la optimización de procesos. Uno de los principales ámbitos de aplicación es la automatización en DevOps, donde modelos híbridos de IA han mostrado una integración eficiente con herramientas de CI/CD ((Senanayake et al., 2024), (Alizadehsani et al., 2023)). Estas soluciones destacan por su capacidad de reducir los tiempos de despliegue y optimizar el flujo de trabajo, facilitando la detección temprana de errores en las primeras etapas de desarrollo. En entornos ágiles, se ha observado una reducción del 40% en los tiempos de entrega, lo que mejora significativamente la velocidad y eficiencia de los procesos de integración y entrega continua.

Otro aspecto clave es la gestión de configuración con IA, donde se han desarrollado modelos avanzados con integración en repositorios y herramientas de versionado (Golshanrad & Faghih, 2024). Estas herramientas permiten la predicción de conflictos y la generación de configuraciones óptimas, lo que es especialmente útil en proyectos de desarrollo a gran escala. Según estudios, la implementación de estos modelos ha logrado una reducción del 30% en

errores de configuración, optimizando la estabilidad y confiabilidad del software. En el ámbito de la revisión de código automatizada, se han aplicado modelos de deep learning para el análisis de calidad del código ((Martins et al., 2023), (Draz et al., 2021)). Estas herramientas tienen la capacidad de detectar código duplicado y sugerir refactorizaciones, lo que mejora la mantenibilidad y legibilidad del software. Su aplicación en proyectos de mantenimiento ha resultado en una disminución del 35% en el tiempo requerido para la revisión de código, permitiendo a los equipos enfocarse en tareas más estratégicas.

Para la estimación de esfuerzo en el SDLC, se han desarrollado modelos predictivos basados en datos históricos y métricas del proyecto ((Nazir et al., 2024)). Estas soluciones pueden predecir con precisión el esfuerzo necesario en función de la complejidad del desarrollo, lo que es fundamental para una mejor planificación y gestión de proyectos. La literatura reporta una precisión del 85% en las estimaciones de esfuerzo, permitiendo a los gerentes de proyectos asignar recursos de manera más efectiva y reducir la incertidumbre en la planificación.

En términos de identificación y prevención de defectos en código fuente, se han implementado algoritmos de detección temprana de errores ((Pauzi & Capiluppi, 2023), (Eramo et al., 2024)). Estas herramientas han demostrado ser altamente eficaces para reducir la cantidad de errores antes de la fase de pruebas, mejorando la calidad del software desde las primeras etapas del desarrollo. Se ha observado que su uso permite una reducción del 25% en defectos en producción, lo que minimiza los costos asociados a correcciones tardías y retrabajos.

La refactorización de código también ha sido un área beneficiada por la IA, con modelos basados en redes neuronales que sugieren reescrituras de código para mejorar su eficiencia y legibilidad ((Tahvili et al., 2020), (Dong et al., 2025)). Estas herramientas han sido utilizadas principalmente en la modernización de aplicaciones, logrando una mejora del 40% en la legibilidad y eficiencia del código, lo que facilita su mantenimiento y evolución a largo plazo.

Finalmente, en el contexto de proyectos colaborativos, se han desarrollado soluciones que emplean machine learning para optimizar la gestión de tareas en equipos de desarrollo distribuidos ((Tameswar et al., 2022)). Estas herramientas analizan patrones de colaboración y sugieren estrategias para mejorar la eficiencia, lo que ha resultado en una optimización del 30% en la asignación de tareas en equipos remotos, fortaleciendo la coordinación y productividad en entornos de trabajo descentralizados.

Los hallazgos amplían el marco comparativo de herramientas de IA en el desarrollo de software, proporcionando a profesionales y empresas una visión más detallada sobre las soluciones disponibles para mejorar la productividad, calidad y eficiencia en diferentes fases del ciclo de vida del software. La comparación de herramientas de inteligencia artificial en el desarrollo de software, basada en la literatura revisada, destaca diferencias clave en términos de eficiencia, adaptabilidad y efectividad en distintas etapas del ciclo de vida del software. Estas diferencias permiten evaluar qué herramientas son más adecuadas según el contexto y los requerimientos específicos de cada proyecto.

En cuanto a la eficiencia, se observa que algunas herramientas sobresalen en tareas específicas de desarrollo. ChatGPT y Codex han demostrado ser altamente eficientes en la generación de código para problemas simples, permitiendo mejoras en productividad y reducción del tiempo de desarrollo. Sin embargo, para problemas más complejos, DeepMind AlphaCode ofrece mejores resultados en la resolución de desafíos algorítmicos avanzados (Kuhail et al., 2024). Además, en el ámbito de la detección de vulnerabilidades y predicción de defectos, los modelos basados en deep learning (DL) aplicados a fuzzing y análisis de código han mostrado incrementos de precisión que varían entre un 7% y un 20%, optimizando la identificación de errores y mejorando la calidad del software ((Bamohabbat Chafjiri et al., 2024), (Ghafoor Hussain et al., 2025), (Mazhar et al., 2024), (Sánchez-García et al., 2023)).

Desde la perspectiva de adaptabilidad, se identifican diferencias en la capacidad de estas herramientas para ajustarse a diversos entornos y necesidades. ChatGPT ha demostrado una gran versatilidad, aplicándose no solo en el desarrollo de software, sino también en ámbitos como la educación y el marketing. Sin embargo, puede presentar limitaciones en la resolución de problemas altamente especializados y complejos. Por otro lado, frameworks como Undulate y Orfeon, así como modelos diseñados para la optimización de pruebas, destacan ((Asikainen & Männistö, 2022), (Mehmood et al., 2024), (Díaz-de-Arcaya et al., 2023)). Estas herramientas pueden adaptarse según el entorno y los requerimientos específicos del proyecto, facilitando la automatización y optimización de procesos en múltiples escenarios.

En términos de efectividad, la literatura muestra que la aplicación de IA en el desarrollo de software ha resultado en mejoras tangibles en reducción de errores, optimización de recursos y aumento en la calidad del producto final. Por ejemplo, los modelos basados en CodeBERT han logrado mejorar significativamente la identificación de defectos en el código, reduciendo los tiempos de corrección y permitiendo una depuración más eficiente ((Ghafoor Hussain et al., 2025), (Mazhar et al., 2024)). En el caso de la generación automática de documentación, GPT-4 ha alcanzado niveles de precisión comparables a los métodos tradicionales, facilitando la elaboración de documentación técnica sin comprometer la calidad (Sivakumar et al., 2024). Asimismo, en el área de predicción del esfuerzo en pruebas, modelos optimizados han demostrado ser particularmente útiles en entornos ágiles, donde la estimación precisa del esfuerzo es crucial para la planificación y gestión eficiente de los proyectos ((Sánchez-García et al., 2023)).

Para facilitar la selección de herramientas de inteligencia artificial en el desarrollo de software, se ha sintetizado un marco comparativo basado en los hallazgos de la literatura. Este marco agrupa las herramientas según facilidad de uso e integración, precisión y rendimiento,

adaptabilidad y escalabilidad, e impacto en la productividad y calidad del software, permitiendo a los profesionales del sector identificar las opciones más adecuadas para sus proyectos.

Este criterio evalúa la interfaz de las herramientas, su facilidad de incorporación en entornos de desarrollo y la curva de aprendizaje necesaria para su adopción.

GitHub Copilot (Codex) destaca por su integración nativa en entornos como Visual Studio Code, permitiendo a los desarrolladores generar código con sugerencias en tiempo real.

ChatGPT, en cambio, ofrece una interfaz conversacional más flexible, útil para la depuración de código y generación de documentación técnica, aunque su uso requiere adaptación a prompts más detallados ((Kuhail et al., 2024), (Ahmad et al., 2023)).

Precisión y Rendimiento

Para medir la efectividad de las herramientas, se consideran métricas como AUC-ROC, mejoras en precisión de predicción de errores y tiempos de respuesta en tareas de generación de código. Modelos de detección de defectos basados en deep learning han logrado mejoras de precisión de hasta un 92% en la predicción de errores en código fuente ((Ghafoor Hussain et al., 2025), (Mazhar et al., 2024), (Sánchez-García et al., 2023).Herramientas como Orfeon y técnicas de predicción de esfuerzo en pruebas han reducido la sobreestimación de trabajo en hasta un 18%, optimizando la planificación ((Díaz-de-Arcaya et al., 2023), (Sánchez-García et al., 2023)).

Adaptabilidad y Escalabilidad

Se analiza la capacidad de las herramientas para ajustarse a distintas fases del ciclo de vida del software y su uso en proyectos de diversas escalas.

Frameworks como Undulate y Orfeon han demostrado una alta adaptabilidad, permitiendo su uso en procesos que van desde la planificación del desarrollo hasta la optimización de pruebas ((Asikainen & Männistö, 2022), (Díaz-de-Arcaya et al., 2023)).

Modelos híbridos en DevOps han mostrado una integración eficiente en pipelines de CI/CD, optimizando tiempos de despliegue en un 40% ((Senanayake et al., 2024), (Alizadehsani et al., 2023)).

Impacto en la Productividad y Calidad del Software

Este aspecto evalúa cómo las herramientas de IA influyen en la reducción de errores, la automatización de procesos y la satisfacción del usuario final. Estudios sobre optimización de pruebas y gestión de errores han registrado reducciones en defectos de hasta un 25% en producción, lo que mejora significativamente la calidad del software ((Mehmood et al., 2024), (Kukkar et al., 2023), (Ikram et al., 2022), (Mazhar et al., 2024), (Babaalla et al., 2024)).

El uso de IA en revisión de código automatizada ha disminuido los tiempos de inspección en un 35%, acelerando la entrega de software sin comprometer su calidad ((Martins et al., 2023), (Draz et al., 2021)).

Este marco, basado en el análisis de 35 estudios científicos y técnicos, proporciona una referencia estructurada para que desarrolladores, gestores de proyectos y entidades interesadas puedan seleccionar la herramienta de IA más adecuada según sus necesidades específicas. Al considerar facilidad de uso, precisión, adaptabilidad y productividad, se facilita la toma de decisiones estratégicas en el desarrollo de software impulsado por IA.

La revisión sistemática demuestra que la inteligencia artificial (IA) está generando un impacto profundo y transformador en el desarrollo de software, mejorando significativamente aspectos como la productividad, la calidad y la seguridad en diversas fases del ciclo de vida del software. A medida que la IA se integra más en las herramientas de desarrollo, se observa un claro patrón hacia la automatización y la mejora de la eficiencia.

Uno de los hallazgos más destacados es el uso de asistentes de codificación como ChatGPT y OpenAI Codex, que aumentan la productividad en tareas de programación hasta en

un 55%. Estos modelos permiten a los desarrolladores centrarse en problemas más complejos y estratégicos, mientras que las herramientas como DeepMind AlphaCode sobresalen en la resolución de desafíos algorítmicos más avanzados. Sin embargo, la selección de la herramienta más adecuada depende del contexto específico y de la naturaleza de la tarea que se está abordando. Por ejemplo, mientras que Codex y ChatGPT son eficaces en tareas repetitivas y simples, AlphaCode destaca en entornos más técnicos y complejos.

En el ámbito de la detección de errores y vulnerabilidades, los modelos de deep learning han demostrado ser una herramienta valiosa. Estas tecnologías han mejorado la tasa de identificación de fallos hasta un 30%, lo que permite una detección más temprana de problemas de seguridad y calidad del software. En particular, herramientas como CodeBERT, con una precisión de hasta el 92% en la identificación de defectos, se consolidan como soluciones robustas y eficientes en el proceso de depuración y mejora continua del software. Esta tendencia de automatización en la detección de fallos está reduciendo el tiempo de validación y aumentando la confianza en la seguridad del producto final.

Asimismo, la IA ha optimizado procesos clave como la toma de decisiones y la planificación. Frameworks como Undulate y Orfeon han permitido una mayor adaptabilidad en las pruebas y la evaluación de alternativas, reduciendo hasta en un 40% el tiempo necesario para estos procesos. Además, los modelos basados en machine learning han logrado mejorar la asignación de recursos humanos en la gestión de requisitos, optimizando la asignación de tareas en un 20%, lo que se traduce en un uso más eficiente del talento dentro de los equipos de desarrollo.

En el campo de la automatización de pruebas, algoritmos como Random Forest y SVM han demostrado su capacidad para aumentar la eficacia de las pruebas de software, con una mejora de hasta un 25%. Esto ha permitido reducir errores y la necesidad de validación manual, lo que mejora la calidad general del producto. En el ámbito de DevOps, los modelos híbridos

aplicados a los pipelines de CI/CD han mejorado en un 40% la eficiencia en la implementación, acelerando los tiempos de entrega y optimizando la calidad durante todo el ciclo de vida del software.

Sin embargo, a pesar de estos avances, existen brechas importantes en la implementación de la IA en el desarrollo de software. Un desafío crítico sigue siendo la explicabilidad de los modelos de IA. Muchos de los modelos de deep learning empleados en el desarrollo de software funcionan como "cajas negras", lo que dificulta la comprensión de cómo toman decisiones o predicciones. Esto genera desconfianza entre los desarrolladores y las organizaciones, quienes requieren claridad sobre el razonamiento detrás de las sugerencias automatizadas para poder tomar decisiones informadas.

Además, los sesgos algorítmicos representan otro desafío significativo. A medida que la IA se integra más profundamente en procesos como la asignación de tareas, la gestión de requisitos y la predicción de defectos, es esencial que los modelos sean evaluados y ajustados para evitar sesgos que puedan afectar la equidad y la calidad del software. El uso de datos históricos para entrenar estos modelos puede incorporar patrones discriminatorios que afectan la imparcialidad en los resultados.

Otra brecha identificada en la literatura es la falta de investigaciones sobre cómo la IA influye en la colaboración entre equipos distribuidos y su impacto a largo plazo en la sostenibilidad del software. Si bien la IA puede mejorar la eficiencia individual y la automatización de tareas, es crucial explorar cómo afecta la dinámica de trabajo en equipos ágiles y cómo puede integrarse de manera efectiva en entornos colaborativos a gran escala.

Conclusión

La inteligencia artificial (IA) ha demostrado ser una herramienta clave en la mejora del desarrollo de software, ofreciendo importantes avances en productividad, calidad y eficiencia.

Herramientas como OpenAI Codex y DeepMind AlphaCode han facilitado la automatización de tareas repetitivas, permitiendo a los desarrolladores centrarse en problemas más complejos. Además, la implementación de modelos de IA, como CodeBERT, ha mejorado significativamente la detección de defectos en el código, alcanzando una precisión de hasta el 92%, lo que contribuye a una mayor seguridad y fiabilidad del software.

Asimismo, la integración de IA en áreas como la gestión de requisitos, la planificación de proyectos y la automatización de pruebas ha demostrado mejorar la asignación de recursos humanos y reducir los tiempos de validación, permitiendo un desarrollo de software más ágil y eficiente. En particular, las mejoras en DevOps y la integración de CI/CD a través de modelos híbridos han optimizado en un 40% los tiempos de despliegue, lo que resulta en una mayor agilidad de los procesos.

Sin embargo, a pesar de estos avances, persisten algunos desafíos importantes. La falta de explicabilidad en los modelos de IA sigue siendo un obstáculo, ya que muchas veces es difícil entender cómo toman decisiones o por qué presentan ciertos resultados. Esto puede generar desconfianza, especialmente en entornos críticos. Además, los sesgos inherentes a los modelos de IA representan un riesgo, ya que pueden influir negativamente en los resultados si no se gestionan adecuadamente.

Por tanto, es esencial seguir investigando en la mejora de la transparencia de los modelos de IA y en la mitigación de sesgos, para asegurar que su adopción se haga de manera ética y eficiente. En resumen, aunque la IA ofrece enormes beneficios para el desarrollo de software, su éxito depende de una integración cuidadosa y de la resolución de los desafíos relacionados con la interpretabilidad y los sesgos.

Referencia Bibliografía

Abbas, S., Aftab, S., Khan, M., Ghazal, T., Hamadi, H., & Yeun, C. (2023). Data and Ensemble Machine Learning Fusion Based Intelligent Software Defect Prediction System. *Computers, Materials & Continua*, *75*(3), 6083–6100. https://doi.org/10.32604/cmc.2023.037933

Ahmad, K., Abdelrazek, M., Arora, C., Agrahari Baniya, A., Bano, M., & Grundy, J. (2023). Requirements engineering framework for human-centered artificial intelligence software systems. *Applied Soft Computing*, *143*, 110455. https://doi.org/10.1016/j.asoc.2023.110455

Alaswad, F., & Poovammal, E. (2022). Software quality prediction using machine learning. *Materials Today: Proceedings*, *62*, 4714–4720. https://doi.org/10.1016/j.matpr.2022.03.165

Alizadehsani, Z., Ghaemi, H., Shahraki, A., Gonzalez-Briones, A., & Corchado, J. M. (2023). DCServCG: A data-centric service code generation using deep learning. *Engineering Applications of Artificial Intelligence*, *123*, 106304. https://doi.org/10.1016/j.engappai.2023.106304

Asikainen, T., & Männistö, T. (2022). Undulate: A framework for data-driven software engineering enabling soft computing. *Information and Software Technology*, *152*, 107039. https://doi.org/10.1016/j.infsof.2022.107039

Babaalla, Z., Abdelmalek, H., Jakimi, A., & Oualla, M. (2024). Extraction of UML class diagrams using deep learning: Comparative study and critical analysis. *Procedia Computer Science*, *236*, 452–459. https://doi.org/10.1016/j.procs.2024.05.053

Bamohabbat Chafjiri, S., Legg, P., Hong, J., & Tsompanas, M.-A. (2024). Vulnerability detection through machine learning-based fuzzing: A systematic review. *Computers & Security*, *143*, 103903. https://doi.org/10.1016/j.cose.2024.103903

Dey, S., & Lee, S.-W. (2021). Multilayered review of safety approaches for machine learning-based systems in the days of AI. *Journal of Systems and Software*, *176*, 110941. https://doi.org/10.1016/j.jss.2021.110941

Díaz-de-Arcaya, J., Torre-Bastida, A. I., Miñón, R., & Almeida, A. (2023). Orfeon: An AIOps framework for the goal-driven operationalization of distributed analytical pipelines. *Future Generation Computer Systems*, *140*, 18–35. https://doi.org/10.1016/j.future.2022.10.008

Dong, X., Wang, J., & Liang, Y. (2025). A Novel Ensemble Classifier Selection Method for Software Defect Prediction. *IEEE Access*, *13*, 25578–25597. IEEE Access. https://doi.org/10.1109/ACCESS.2025.3537658

Draz, M., Farhan, M., Abdulkader, S., & Gafar, M. (2021). Code Smell Detection Using Whale Optimization Algorithm. *Computers, Materials & Continua*, *68*(2), 1919–1935. https://doi.org/10.32604/cmc.2021.015586

Eramo, R., Said, B., Oriol, M., Bruneliere, H., & Morales, S. (2024). An architecture for model-based and intelligent automation in DevOps. *Journal of Systems and Software*, *217*, 112180. https://doi.org/10.1016/j.jss.2024.112180

Faruqui, N., Thatoi, P., Choudhary, R., Roncevic, I., Alqahtani, H., Sarker, I. H., & Khanam, S. (2024). AI-Analyst: An AI-Assisted SDLC Analysis Framework for Business Cost Optimization. *IEEE Access*, *12*, 195188–195203. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3519423

Ghafoor Hussain, R., Yow, K.-C., & Gori, M. (2025). Leveraging an Enhanced CodeBERT-Based Model for Multiclass Software Defect Prediction via Defect Classification. *IEEE Access*, *13*, 24383–24397. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3525069

Gill, S. S., Tuli, S., Xu, M., Singh, I., Singh, K. V., Lindsay, D., Tuli, S., Smirnova, D., Singh, M., Jain, U., Pervaiz, H., Sehgal, B., Kaila, S. S., Misra, S., Aslanpour, M. S., Mehta, H., Stankovski, V., & Garraghan, P. (2019). Transformative effects of IoT,

Blockchain and Artificial Intelligence on cloud computing: Evolution, vision, trends and open challenges. *Internet of Things*, *8*, 100118. https://doi.org/10.1016/j.iot.2019.100118

Golshanrad, P., & Faghih, F. (2024). DeepCover: Advancing RNN test coverage and online error prediction using state machine extraction. *Journal of Systems and Software*, *211*, 111987. https://doi.org/10.1016/j.jss.2024.111987

Ikram, A., Jalil, M., Ngah, A., Raza, S., Khan, A., Mahmood, Y., Kama, N., Azmi, A., & Alzayed, A. (2022). Project Assessment in Offshore Software Maintenance Outsourcing Using Deep Extreme Learning Machines. *Computers, Materials & Continua*, *74*(1), 1871–1886. https://doi.org/10.32604/cmc.2023.030818

Kong, L. S., Jasser, M. B., Ajibade, S.-S. M., & Mohamed, A. W. (2024). A systematic review on software reliability prediction via swarm intelligence algorithms. *Journal of King Saud University - Computer and Information Sciences*, *36*(7), 102132. https://doi.org/10.1016/j.jksuci.2024.102132

Kuhail, M. A., Mathew, S. S., Khalil, A., Berengueres, J., & Shah, S. J. H. (2024). “Will I be replaced?” Assessing ChatGPT’s effect on software development and programmer perceptions of AI tools. *Science of Computer Programming*, *235*, 103111. https://doi.org/10.1016/j.scico.2024.103111

Kukkar, A., Kumar Lilhore, U., Frnda, J., Kaur Sandhu, J., Prava Das, R., Goyal, N., Kumar, A., Muduli, K., & Rezac, F. (2023). ProRE: An ACO- based programmer recommendation model to precisely manage software bugs. *Journal of King Saud*

University - Computer and Information Sciences, *35*(1), 483–498. https://doi.org/10.1016/j.jksuci.2022.12.017

Kumar, G., Abubakar Imam, A., Basri, S., Sobri Hashim, A., Ghani Haji Naim, A., Fernando Capretz, L., Balogun, A. O., & Mamman, H. (2024). Ensemble Balanced Nested Dichotomy Fuzzy Models for Software Requirement Risk Prediction. *IEEE Access*, *12*, 146225–146243. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3473942

Lopes, F., Agnelo, J., Teixeira, C. A., Laranjeiro, N., & Bernardino, J. (2020). Automating orthogonal defect classification using machine learning algorithms. *Future Generation Computer Systems*, *102*, 932–947. https://doi.org/10.1016/j.future.2019.09.009

Martins, J., Branco, F., & Mamede, H. (2023). Combining low-code development with ChatGPT to novel no-code approaches: A focus-group study. *Intelligent Systems with Applications*, *20*, 200289. https://doi.org/10.1016/j.iswa.2023.200289

Mazhar, Q., Arif, F., Aurangzeb, K., Ain, N., Khan, J., Rubab, S., & Anwar, M. (2024). Identification of Software Bugs by Analyzing Natural Language-Based Requirements Using Optimized Deep Learning Features. *Computers, Materials & Continua*, *78*(3), 4379–4397. https://doi.org/10.32604/cmc.2024.047172

Mehmood, A., Ilyas, Q. M., Ahmad, M., & Shi, Z. (2024). Test Suite Optimization Using Machine Learning Techniques: A Comprehensive Study. *IEEE Access*, *12*, 168645–168671. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3490453

Nazir, R., Bucaioni, A., & Pelliccione, P. (2024). Architecting ML-enabled systems: Challenges, best practices, and design decisions. *Journal of Systems and Software*, *207*, 111860. https://doi.org/10.1016/j.jss.2023.111860

Nehzati, M. (2024). Integrating convolutional neural networks for improved software engineering: A Collaborative and unbalanced data Perspective. *Memories - Materials, Devices, Circuits and Systems*, *8*, 100106. https://doi.org/10.1016/j.memori.2024.100106

Pauzi, Z., & Capiluppi, A. (2023). Applications of natural language processing in software traceability: A systematic mapping study. *Journal of Systems and Software*, *198*, 111616. https://doi.org/10.1016/j.jss.2023.111616

Perkusich, M., Chaves e Silva, L., Costa, A., Ramos, F., Saraiva, R., Freire, A., Dilorenzo, E., Dantas, E., Santos, D., Gorgônio, K., Almeida, H., & Perkusich, A. (2020). Intelligent software engineering in the context of agile software development: A systematic literature review. *Information and Software Technology*, *119*, 106241. https://doi.org/10.1016/j.infsof.2019.106241

Sánchez-García, A. J., López-Martín, C., & Abran, A. (2023). Gradient Boosting Optimized Through Differential Evolution for Predicting the Testing Effort of Software Projects. *IEEE Access*, *11*, 135235–135254. IEEE Access. https://doi.org/10.1109/ACCESS.2023.3337809

Senanayake, J., Kalutarage, H., Petrovski, A., Piras, L., & Al-Kadri, M. O. (2024). Defendroid: Real-time Android code vulnerability detection via blockchain federated

neural network with XAI. *Journal of Information Security and Applications*, *82*, 103741. https://doi.org/10.1016/j.jisa.2024.103741

Sivakumar, M., Belle, A. B., Shan, J., & Khakzad Shahandashti, K. (2024). Prompting GPT –4 to support automatic safety case generation. *Expert Systems with Applications*, *255*, 124653. https://doi.org/10.1016/j.eswa.2024.124653

Tahvili, S., Hatvani, L., Ramentol, E., Pimentel, R., Afzal, W., & Herrera, F. (2020). A novel methodology to classify test cases using natural language processing and imbalanced learning. *Engineering Applications of Artificial Intelligence*, *95*, 103878. https://doi.org/10.1016/j.engappai.2020.103878

Tameswar, K., Suddul, G., & Dookhitram, K. (2022). A hybrid deep learning approach with genetic and coral reefs metaheuristics for enhanced defect detection in software. *International*

Journal of Information Management Data Insights, *2*(2), 100105. https://doi.org/10.1016/j.jjimei.2022.100105

Wang, B., Zou, Z., Wan, H., Li, Y., Deng, Y., & Li, X. (2024). An empirical study on the state-of-the-art methods for requirement-to-code traceability link recovery. *Journal of King Saud University - Computer and Information Sciences*, *36*(6), 102118. https://doi.org/10.1016/j.jksuci.2024.102118