



**UNIVERSIDAD
NACIONAL
DE LA PLATA**



**"Trabajo Final presentado para obtener el grado de Especialista
en Ingeniería de Software"**

**"Desde los Obstáculos hasta las
Estrategias: Integración de la
inteligencia artificial en el ciclo de vida
del desarrollo del software."**

ALUMNA: Lic. Marcela Paladino

DIRECTORA: Dra. Claudia Pons

ABRIL 2025



ÍNDICE

RESUMEN	4
Capítulo 1: INTRODUCCIÓN	5
1.1 Contexto y Relevancia del Tema	5
1.2 Planteamiento del Problema	5
1.3 Pregunta de investigación	6
1.4 Objetivos de la investigación	6
1.4.1 Objetivo general	6
1.4.2 Objetivos específicos	6
1.5 Justificación	7
1.6 Estructura del Documento	7
Capítulo 2: MARCO TEÓRICO	10
2.1 Inteligencia Artificial: Revolución en el Desarrollo de Software	10
2.2 Desafíos Éticos y Técnicos en la Integración de la IA	11
2.3 Hacia una Integración Exitosa: Estrategias y Modelos	11
Capítulo 3: METODOLOGÍA	12
3.1 Tipo de Investigación	12
3.2 Criterios de Selección de Artículos	12
3.2.1 Criterios de inclusión	12
3.2.2 Criterios de exclusión	12
3.3 Proceso de Búsqueda y Selección de la Literatura	13
3.4 Herramientas y Enfoques de Análisis	14
Capítulo 4: RESULTADOS	14
4.1 Clasificación y Aplicación de Herramientas de AI en el SDLC	14
4.2 Revisión y síntesis de la información extraída: Clasificación de las Herramientas de AI por Etapa del SDLC	18
4.3 Desafíos, Factores Contribuyentes, Estrategias Sugeridas e Impacto en el SDLC	19
4.3.1 Temáticas y Desafíos Recurrentes	20
4.3.2 Estrategias y Recomendaciones Identificadas	21
4.3.3 Detección de Tendencias y Vacíos de la Literatura	21
4.3.4 Identificación de las Etapas Críticas del SDLC en la Integración de IA	22
4.3.5 Exploración de los Factores Técnicos, Organizacionales y Éticos que Dificultan la Adopción de AI en el SDLC	23



UNIVERSIDAD
NACIONAL
DE LA PLATA



**“Desde los Obstáculos hasta las Estrategias: Integración
de la inteligencia artificial en el ciclo de vida del
desarrollo del software.”**

4.3.6 Ejemplos Prácticos y Estudios de Caso en la Integración de AI en el SDLC	26
4.4 Análisis de la Calidad de los Artículos Seleccionados	27
Capítulo 5: ANALISIS Y DISCUSIÓN	30
5.1 Desafíos Técnicos: Entre el Potencial y las Limitaciones	30
5.2 Barreras Organizacionales y Transformación Cultural	31
5.3 Consideraciones Éticas y el Desafío de la Transparencia	32
5.4 Síntesis de la Controversia y Proyecciones Futuras	32
5.5 Recomendaciones	33
5.6 Limitaciones del Estudio	35
Capítulo 6: CONCLUSIÓN	35
REFERENCIAS BIBLIOGRAFICAS	38
ANEXOS	41



RESUMEN

En el presente estudio se realizó una revisión narrativa sobre la integración de la AI (artificial intelligence) en el SDLC (Software Development Life Cycle), analizando tanto los avances tecnológicos como los obstáculos que impiden su adopción plena. A través de una sistematización de la información extraída de estudios recientes, se identificaron y clasificaron las principales herramientas de AI aplicadas en cada fase del SDLC, así como los desafíos asociados, que se agrupan en dimensiones técnicas, organizacionales y éticas. La metodología empleada incluyó la revisión y análisis comparativo de literatura de alta relevancia, utilizando criterios de selección rigurosos y la elaboración de una matriz de extracción de datos para identificar patrones y tendencias en la integración de la AI. Los hallazgos revelan que, aunque la aplicación de técnicas como el aprendizaje profundo, el procesamiento de lenguaje natural y los agentes inteligentes ofrecen oportunidades para optimizar procesos clave (como la planificación, el diseño, la generación de código, las pruebas y el mantenimiento), persisten barreras significativas que limitan la efectividad de estas estrategias. Entre los desafíos se destacan la calidad y la estructuración de datos, la compatibilidad con sistemas heredados, la resistencia organizacional y la falta de transparencia en los modelos de AI. En consecuencia, se proponen estrategias integrales que combinan mejoras técnicas, amplios programas de capacitación y marcos éticos robustos para favorecer una integración más sostenible y eficiente de la AI.



Capítulo 1: INTRODUCCIÓN

1.1 Contexto y Relevancia del Tema

La integración de la AI en el SDLC representa una de las tendencias más disruptivas en la ingeniería de software de la última década. Con el auge de técnicas como el aprendizaje profundo, el procesamiento de lenguaje natural y diversos algoritmos de optimización, la AI no solo ha permitido automatizar tareas rutinarias, sino que también ha posibilitado mejoras significativas en la eficiencia y precisión de procesos críticos, desde la planificación y el análisis hasta la implementación, las pruebas y el mantenimiento (Benitez & Serrano, 2023; Durrani et al., 2024).

En un contexto de creciente competitividad y transformación digital, la aplicación de la AI se ha convertido en un elemento estratégico para responder a la demanda del mercado global. La capacidad de interpretar grandes volúmenes de datos y transformar esa información en decisiones automatizadas está impulsando una nueva era en la que el desarrollo de software se orienta hacia modelos más ágiles y adaptativos. Esta tendencia ha generado un creciente interés tanto en la academia como en la industria, evidenciando la importancia de comprender el impacto y las implicaciones de la AI en cada fase del SDLC. Así, se plantea la necesidad de revisar y sintetizar el estado del arte, para consolidar las oportunidades que la AI ofrece y para identificar de manera clara los desafíos que persisten en su adopción.

1.2 Planteamiento del Problema

Aunque la integración de la AI en el SDLC ofrece un potencial transformador para mejorar la productividad y la calidad del software, su adopción plena aún se ve obstaculizada por múltiples barreras.

En primer lugar, los desafíos técnicos son particularmente notorios: la efectividad de los modelos de AI depende en gran medida de la disponibilidad de datos limpios y estructurados. Autores como Giffari et al. (2024) y Müller (2024) han señalado que la presencia de datos fragmentados o de baja calidad reduce la confiabilidad de estos sistemas, mientras que la falta de transparencia en el funcionamiento de muchos algoritmos dificulta su explicabilidad, generando incertidumbre en la interpretación de sus resultados (Özpolat et al., 2023; Siddique, 2022).



Por otro lado, la integración de soluciones basadas en AI con sistemas tradicionales presenta un reto significativo. Los entornos de desarrollo que operan con arquitecturas heredadas y flujos de trabajo consolidados requieren ajustes técnicos sustanciales para incorporar nuevas herramientas, tal como indican Haghsheno (2020) y Ajiga et al. (2024). Esta incompatibilidad puede implicar reestructuraciones complejas y costosas, afectando tanto la implementación como el mantenimiento de los proyectos de software.

Además, en el ámbito organizacional y ético se evidencian otros obstáculos. La resistencia al cambio, derivada del temor a una posible reducción de la relevancia de las habilidades tradicionales o incluso el desplazamiento de roles laborales, es una barrera importante (Haghsheno, 2020; Alenezi & Akour, 2025). Sumado a ello, surgen preocupaciones sobre la privacidad de los datos, la equidad en los algoritmos y la transparencia en la toma de decisiones automatizadas, aspectos que insistentes estudios (Todorov, 2022; Russo, 2024) han destacado como fundamentales para ganar la confianza de desarrolladores y usuarios.

En conjunto, estas barreras, técnicas, de integración, organizacionales y éticas, conforman un escenario complejo que limita el impacto positivo que la AI puede tener en la transformación digital de los procesos de desarrollo de software. Este problema constituye el núcleo del presente estudio, orientado a identificar, analizar y, eventualmente, proponer estrategias que permitan superar dichos obstáculos.

1.3 Pregunta de investigación

"¿Cuáles son los principales desafíos y barreras en la integración de la inteligencia artificial en el ciclo de vida del desarrollo de software?"

1.4 Objetivos de la investigación

1.4.1 Objetivo general

Analizar los principales desafíos y barreras en la integración de la AI en el SDLC, identificando su impacto en cada etapa y proponiendo estrategias para superarlos.

1.4.2 Objetivos específicos

1. **Identificar** las etapas del SDLC donde se presentan los mayores desafíos y barreras en la integración de IA.



2. **Explorar** los factores técnicos, organizacionales y éticos que dificultan la adopción de AI en las distintas etapas del SDLC.
3. **Examinar** estudios de caso o ejemplos prácticos que evidencien los desafíos recurrentes en proyectos que implementan AI en el desarrollo de software.
4. **Proponer** estrategias fundamentadas para superar los desafíos y barreras identificados, promoviendo una integración exitosa de la AI en el SDLC.

1.5 Justificación

El abordaje de los desafíos en la integración de la AI en el SDLC es crucial, tanto para aprovechar las oportunidades de innovación como para garantizar el desarrollo de software de alta calidad, seguro y eficiente. La literatura es amplia, pero aún presenta una dispersión en el análisis de los factores técnicos, organizativos y éticos que inciden en la adopción de la AI (Pothukuchi et al., 2023; Russo, 2024). Este vacío en el conocimiento dificulta la formulación de estrategias integrales que permitan a las organizaciones transformar sus procesos de desarrollo de manera sostenible.

En cuanto a la delimitación del problema, este estudio se enfoca en la integración de soluciones de AI en entornos de desarrollo de software tradicionales, abarcando específicamente las fases de planificación, análisis, desarrollo, pruebas y mantenimiento. Este enfoque permite evitar una dispersión temática y concentrarse en los aspectos que tienen un impacto directo en la calidad, eficiencia y seguridad del software producido.

En resumen, atender de manera integral los retos que dificultan la plena adopción de la AI en el SDLC es esencial para maximizar sus beneficios. Este estudio busca ofrecer información relevante que permita tener una estrategia para la implementación responsable de tecnologías de IA, alineada con las exigencias contemporáneas de la industria del software.

1.6 Estructura del Documento

Este trabajo se organiza en seis capítulos principales, complementados por las referencias bibliográficas y los anexos. A continuación, se detalla el contenido de cada sección:

Capítulo 1: Introducción



- **Contexto y Relevancia del Tema:** Se establece el panorama actual en el que la AI está transformando el desarrollo de software, resaltando la importancia de abordar sus desafíos y oportunidades.
- **Planteamiento del Problema y Pregunta de Investigación:** Se definen los obstáculos y barreras que dificultan la adopción plena de la AI en el SDLC y se formula la pregunta central que orienta el estudio.
- **Objetivos de la Investigación:** Se presentan los objetivos generales y específicos, orientados a identificar, analizar y proponer estrategias para superar los desafíos en la integración de IA.
- **Justificación:** Se expone la relevancia académica y práctica del estudio, subrayando la necesidad de una aproximación holística que abarque aspectos técnicos, organizativos y éticos.
- **Estructura del Documento:** Se describe brevemente la organización de los capítulos siguientes, ofreciendo una hoja de ruta al lector.

Capítulo 2: Metodología

- **Enfoque Metodológico:** Se establece el uso de la revisión narrativa como método para analizar la literatura y sintetizar evidencias sobre la integración de la AI en el SDLC.
- **Criterios de Selección y Fuentes de Información:** Se detallan los criterios de inclusión y exclusión utilizados para elegir los estudios, así como las principales bases de datos consultadas.
- **Procedimiento de Extracción y Sistematización de Datos:** Se explica el proceso de elaboración de la matriz de extracción de información, la codificación temática y el análisis comparativo de los hallazgos.
- **Estrategias de Análisis:** Se describen las técnicas empleadas para sintetizar y contrastar los desafíos, estrategias y resultados presentados en la literatura.

Capítulo 3: Marco Teórico



- **Conceptos Fundamentales de AI y el SDLC:** Se definen y contextualizan los conceptos clave relacionados con la inteligencia artificial (aprendizaje profundo, NLP, algoritmos de optimización, etc.) y se describe el ciclo de vida del desarrollo de software.
- **Intersección entre AI y Desarrollo de Software:** Se analiza cómo la AI ha sido aplicada en cada fase del SDLC, destacando beneficios, oportunidades y desafíos.
- **Revisión de Estudios Previos y Teorías Relevantes:** Se sintetizan los principales enfoques, modelos y debates teóricos en torno a la integración de la AI en la ingeniería de software.

Capítulo 4: Resultados y Análisis

- **Presentación de Hallazgos:** Se exponen los resultados obtenidos a partir de la revisión de la literatura, organizados según las fases del SDLC (planificación, recolección de requisitos, análisis, diseño, desarrollo, pruebas, despliegue y mantenimiento).
- **Clasificación de Herramientas y Desafíos:** Se describen las principales herramientas de AI identificadas, los obstáculos enfrentados (técnicos, organizacionales y éticos) y las estrategias sugeridas para superarlos.
- **Análisis Comparativo:** Se contrastan los hallazgos de diferentes estudios, se identifican patrones comunes y se evidencian vacíos en la literatura.

Capítulo 5: Discusión

- **Interpretación de los Resultados:** Se analizan en profundidad los desafíos identificados, relacionándolos con los marcos teóricos y los estudios previos.
- **Implicaciones para la Práctica y la Investigación:** Se discuten las repercusiones de los hallazgos en el ámbito práctico (implementación en entornos reales) y se proponen áreas para investigaciones futuras.
- **Limitaciones del Estudio:** Se reconocen las limitaciones inherentes al enfoque metodológico de la revisión narrativa y se plantea cómo estas limitaciones podrían abordarse en investigaciones subsiguientes.

Capítulo 6: Conclusiones y Recomendaciones



- **Síntesis de Conclusiones:** Se resumen los principales hallazgos del estudio, respondiendo a la pregunta de investigación.
- **Recomendaciones Prácticas:** Se proponen estrategias y acciones concretas que faciliten una integración más efectiva y sostenible de la AI en el SDLC.

Referencias bibliográficas

- Se incluye un listado completo de las fuentes y referencias consultadas durante el proceso de investigación en formato APA (American Psychological Association).

Anexos

- Material adicional que respalda el estudio, como la matriz de extracción de datos, tablas de evaluación de la calidad de los estudios y otros documentos relacionados con la sistematización y análisis de la información.

Capítulo 2: MARCO TEÓRICO

2.1 Inteligencia Artificial: Revolución en el Desarrollo de Software

La AI ha surgido como un pilar en la evolución del desarrollo de software, redefiniendo cómo los equipos tecnológicos enfrentan los desafíos complejos y optimizan los procesos. La integración de AI en el SDLC ha sido reconocida como una innovación clave que amplifica tanto la automatización como la capacidad de análisis estratégico en cada etapa del ciclo.

En el contexto del análisis de requisitos y modelado, Simon et al. (2021), enfatizan cómo los algoritmos de aprendizaje profundo pueden identificar patrones ocultos en grandes volúmenes de datos, reduciendo significativamente los errores iniciales en proyectos. Complementariamente, Zhang (2020), destaca el impacto de modelos generativos en la fase de diseño, facilitando la creación de prototipos interactivos que mejoran la colaboración entre desarrolladores y stakeholders.

Por otro lado, Alford (2019), explora cómo los agentes inteligentes pueden automatizar pipelines de DevOps, permitiendo entregas continuas más ágiles y reduciendo la intervención humana en tareas repetitivas. Este enfoque no solo mejora la eficiencia en la implementación, sino que también abre la puerta a metodologías más adaptativas en la gestión de proyectos.



Sin embargo, como señala Brown et al. (2018), la adopción de la AI en estas fases requiere que las organizaciones ajusten sus dinámicas internas, enfrentando barreras organizativas y la resistencia al cambio entre los equipos de desarrollo.

En términos de pruebas y aseguramiento de calidad, Kumar (2017), propone la integración de sistemas híbridos que combinan supervisión humana con herramientas automatizadas basadas en AI. Estas estrategias permiten abordar los desafíos de precisión y fiabilidad en entornos dinámicos, especialmente cuando se evalúan productos de software destinados a mercados altamente regulados.

2.2 Desafíos Éticos y Técnicos en la Integración de la IA

Además de los beneficios, los retos asociados con la implementación de herramientas de AI en el SDLC han sido extensamente documentados en la literatura. García et al. (2022), resaltan que las preocupaciones sobre privacidad de datos y sesgos algorítmicos representan barreras fundamentales para la confianza en estos sistemas. Müller (2020) complementa esta discusión señalando que la falta de transparencia en modelos de AI dificulta su aceptación entre los desarrolladores, quienes suelen desconfiar de herramientas con dinámicas de “caja negra”.

En el ámbito técnico, Lee y Park (2015), proponen marcos teóricos como HACAF para analizar las interacciones entre herramientas basadas en AI y sistemas heredados. Estos enfoques teóricos permiten identificar puntos de fricción en la integración y establecer pautas para resolver incompatibilidades con arquitecturas tradicionales.

2.3 Hacia una Integración Exitosa: Estrategias y Modelos

La literatura también propone estrategias para superar estos desafíos y maximizar el impacto de la AI en el SDLC. Keskar (2023), sugiere combinar herramientas de AI generativa como GitHub Copilot con plataformas de colaboración en equipo para fomentar un ambiente de desarrollo más integrado. Además, Kumar (2017) subraya la importancia de invertir en programas de capacitación para los equipos de desarrollo, mitigando la resistencia organizacional y mejorando la adaptabilidad tecnológica.

Finalmente, estudios como el de Oluwaseun (2023) hacen hincapié en el potencial de la AI para abordar la calidad del producto final, especialmente en sectores críticos como el de la salud. Según el autor, la AI no solo puede acelerar los procesos de validación, sino también garantizar



resultados que cumplan con normativas específicas, promoviendo una adopción ética y sostenible.

Capítulo 3: METODOLOGÍA

3.1 Tipo de Investigación

Esta investigación adopta un enfoque de revisión narrativa. El objetivo es sintetizar y analizar críticamente, a través de estudios, la literatura existente sobre los desafíos y barreras en la integración de la inteligencia artificial (IA) en el ciclo de vida del desarrollo de software (SDLC). La revisión narrativa resulta especialmente útil para explorar tendencias, identificar vacíos y elaborar propuestas estratégicas a partir de la evidencia empírica disponible.

3.2 Criterios de Selección de Artículos

Para asegurar la relevancia y calidad de la muestra de estudios incluidos en esta revisión, se establecieron los siguientes criterios de inclusión y exclusión:

3.2.1 Criterios de inclusión:

- Artículos académicos publicados entre 2022 y 2025.
- Estudios que aborden de forma directa la integración de la AI en las diversas fases del SDLC, tales como planificación, diseño, desarrollo, pruebas, despliegue y mantenimiento.
- Publicaciones en bases de datos reconocidas, principalmente en Google Scholar (fuente predominante), pero también en IEEE Xplore, Scopus, ACM Digital Library y Web of Science.
- Investigaciones disponibles en inglés o español.
- Estudios que presenten análisis de desafíos, barreras o estrategias relacionadas con la implementación de la AI en el SDLC.

3.2.2 Criterios de exclusión:

- Publicaciones cuyos resultados no resulten claros o relevantes para la pregunta de investigación.



- Estudios que se enfoquen únicamente en aspectos técnicos de la AI sin considerar su integración global en el SDLC.
- Documentos cuya única disponibilidad sea en forma de resumen o que tengan acceso restringido.

3.3 Proceso de Búsqueda y Selección de la Literatura

El proceso se desarrolló en varias etapas para garantizar un enfoque exhaustivo y riguroso:

1. **Estrategia de búsqueda (definición de palabras clave y operadores booleanos):** Se utilizaron combinaciones de términos como: ("artificial intelligence" OR AI) AND ("software lifecycle" OR "software development lifecycle") AND (challenges OR barriers OR obstacles) AND (integration OR adoption OR implementation). Estas combinaciones permitieron obtener un amplio espectro de estudios relacionados.
2. **Fuentes de Información:** Aunque se consultaron diversas bases de datos (IEEE Xplore, Scopus, ACM Digital Library y Web of Science), la mayor parte de los artículos fueron descargados de Google Scholar, lo cual facilitó el acceso a una amplia variedad de estudios académicos y técnicos.
3. **Revisión Inicial de Títulos y Resúmenes:** Se realizó un primer filtro mediante la revisión de títulos y resúmenes, eliminando aquellos estudios que no cumplían con los criterios de inclusión definidos.
4. **Aplicación de Criterios de Inclusión y Exclusión:** Tras el filtrado inicial, se llevó a cabo una lectura detallada para asegurar que los artículos seleccionados aportaran información relevante y que sus resultados fueran claros en relación a la integración de la AI en el SDLC.
5. **Evaluación de Calidad:** Se aplicó una matriz de evaluación de calidad, en la cual se valoró la claridad, relevancia y rigor metodológico de cada estudio. Este proceso permitió priorizar aquellos artículos que ofrecían una contribución sustancial y confiable a la temática en estudio.
6. **Organización de la Información:** La información relevante se sistematizó en una hoja de cálculo (la matriz de extracción de datos), donde se clasificaron los artículos



considerando las fases del SDLC a que hacían referencia, los desafíos identificados, los factores contributivos, las estrategias sugeridas y su impacto en la eficiencia y calidad del software.

3.4 Herramientas y Enfoques de Análisis

Para analizar y sintetizar la información, se emplearon diversos métodos y herramientas:

1. **Síntesis Narrativa:** Se elaboraron narrativas integradoras para resumir los hallazgos de los estudios, identificando patrones, tendencias y vacíos en la literatura. Este enfoque permitió contextualizar los resultados en función de los objetivos de la investigación.
2. **Matriz de Análisis Comparativo:** Se construyó una matriz de extracción de datos (Anexo 2) que compila la información clave de 30 artículos académicos. Esta matriz agrupa los desafíos, factores contributivos, estrategias y el impacto reportado, ofreciendo una estructura clara para identificar patrones y tendencias.
3. **Codificación Temática:** Se aplicó un análisis cualitativo basado en la codificación temática, identificando categorías recurrentes en los textos seleccionados. Esto permitió clasificar los datos en torno a desafíos técnicos, organizacionales y éticos, facilitando la interpretación de los hallazgos.

Este enfoque metodológico garantiza un análisis riguroso y estructurado de la literatura, permitiendo abordar de manera integral los retos asociados a la integración de la AI en el SDLC. La combinación de una revisión narrativa, la sistematización mediante una matriz de extracción de datos y el uso de herramientas de análisis cualitativo proporciona fundamentos sólidos para la discusión y conclusiones del trabajo.

Capítulo 4: RESULTADOS

4.1 Clasificación y Aplicación de Herramientas de AI en el SDLC

En la presente sección se exponen los resultados obtenidos del análisis de la literatura, destacándose las principales herramientas de AI aplicadas en cada etapa del SDLC. A través de esta revisión se identificaron las tecnologías más relevantes, detallando las fases específicas del SDLC en las que son utilizadas, así como las fuentes bibliográficas que respaldan su implementación.



Los resultados de estudios destacaron cómo la AI se ha integrado de manera específica en procesos como la planificación, recolección de requisitos, análisis, diseño, desarrollo, pruebas, despliegue y mantenimiento. Además, se evidenciaron tendencias y enfoques diferenciados que responden a los desafíos particulares de cada fase.

Si bien, los resultados generales de las herramientas de AI utilizadas en el SDLC se detallan en el Anexo 1, a continuación se presentan las mismas agrupadas según su aplicación en las diferentes fases. Esta estructura busca proporcionar una visión comprensiva de la diversidad y el enfoque de las herramientas de AI reportadas en la literatura, estableciendo así una base sólida para el análisis comparativo y la discusión de los resultados posteriores.

Planificación

La fase de planificación utiliza herramientas avanzadas de AI para optimizar la estimación de recursos, predecir cronogramas y organizar proyectos:

- **OCR (Reconocimiento Óptico de Caracteres):** Extrae texto de imágenes o archivos PDF, facilitando la generación de requisitos iniciales (Giffari et al., 2024).
- **Modelos de aprendizaje profundo:** Predicen modelos del SDLC y estiman el tamaño del software basado en puntos de función (Benitez & Serrano, 2023; Durrani et al., 2024; Khan, Fatima & Ahmed, 2023).
- **Agentes inteligentes:** Automatizan la planificación de tareas y la asignación de recursos en pipelines DevOps (Kambala, 2024).
- **Modelos LSTM y RNN:** Usados para predicciones de cronogramas y asignación de recursos (Kumar, 2024).
- **Plataformas de gestión como JIRA Automation y Monday.com:** Ayudan con estimaciones y seguimiento del progreso del proyecto (Shah & Trehan, 2024; Alenezi & Akour, 2025; Zhang, 2024).
- **ChatGPT:** Genera documentación inicial y organiza la definición de requisitos (Ricárdez et al., 2024; Russo, 2024).

Recolección de Requisitos



En esta etapa, la AI facilita la extracción, estructuración y análisis de requisitos:

- **Procesamiento de lenguaje natural (NLP):** Analiza documentos para estructurar los requisitos y definir el alcance del proyecto (Garg, 2023; Kumar, 2024; Siddique, 2022; Sergeyuk et al., 2025).
- **ChatGPT:** Interpreta requisitos textuales complejos y asiste en la generación de ideas iniciales (Özpolat et al., 2023; Ricárdez et al., 2024).
- **Machine Learning:** Analiza datos históricos y patrones para definir necesidades específicas del software (Durrani et al., 2024; Pothukuchi et al., 2023).

Diseño

En el diseño del software, las herramientas de AI ayudan a crear arquitecturas, diagramas y patrones eficientes:

- **Sketch2Code:** Convierte bocetos en código HTML automáticamente (Upadhyaya, 2022; Kokol, 2024; Sergeyuk et al., 2025).
- **Generative AI:** Diseña patrones arquitectónicos y mockups UX (Keskar, 2023; Medepalli, 2025; Pothukuchi et al., 2023).
- **Simuladores basados en IA:** Evalúan arquitecturas y diseños adaptativos, asegurando su funcionalidad (Oluwaseun, 2023; Shao et al., 2025).
- **Adobe XD y ArchiMate AI:** Facilitan la optimización y validación de diseños arquitectónicos (Kokol, 2024; Alenezi & Akour, 2025).

Implementación

La implementación se revoluciona con herramientas generativas y asistentes inteligentes:

- **Generadores automáticos de código:** Herramientas como GitHub Copilot, OpenAI Codex y Facebook TransCoder generan código y traducen entre lenguajes (Müller, 2024; Ricárdez et al., 2024; Özpolat et al., 2023).
- **Clasificación de errores:** Machine Learning detecta vulnerabilidades en tiempo real (Benitez & Serrano, 2023; Kambala, 2024; Ajiga et al., 2024).



- **Asistentes para autocompletado:** Ayudan a los desarrolladores con funciones específicas y sugerencias contextuales (Upadhyaya, 2022; Kumar, 2024).

Pruebas

La fase de pruebas incorpora AI para automatizar y mejorar la validación de software:

- **Deep Learning y NLP:** Generan y clasifican escenarios de prueba (Garg, 2023; Khan, Fatima & Ahmed, 2023; Durrani et al., 2024).
- **Frameworks inteligentes:** Testim, Test.AI y Applitools optimizan las pruebas y detectan fallas (Shah & Trehan, 2024; Medepalli, 2025).
- **Herramientas predictivas:** Supervisan las pruebas y anticipan errores críticos (Müller, 2024; Haghsheno, 2020; Pothukuchi et al., 2023).

Despliegue

Durante el despliegue, las herramientas de AI aseguran estabilidad y optimización:

- **Sistemas auto-adaptativos:** Aprendizaje por refuerzo que ajusta recursos dinámicamente (Müller, 2024; Haghsheno, 2020; Kumar, 2024).
- **Pipelines CI/CD:** Jenkins AI y Automation Anywhere integran y despliegan software automáticamente (Madupati, 2025; Vadde & Munagandla, 2022).
- **Monitoreo predictivo:** Dynatrace y New Relic analizan rendimiento post-despliegue (Kokol, 2024; Upadhyaya, 2022; Sergeyuk et al., 2025).

Mantenimiento

El mantenimiento del software utiliza herramientas predictivas y de análisis proactivo:

- **Modelos predictivos:** Detectan y analizan deuda técnica para mantener la estabilidad del sistema (Benitez & Serrano, 2023; Ajiga et al., 2024; Garg, 2023).
- **Refactorización automática:** Reparación de código utilizando algoritmos de búsqueda semántica (Haghsheno, 2020; Müller, 2024).

- **Monitoreo continuo:** AI supervisa el sistema en tiempo real y realiza ajustes automáticos (Medepalli, 2025; Upadhyaya, 2022; Oluwaseun, 2023).

4.2 Revisión y síntesis de la información extraída: Clasificación de las Herramientas de AI por Etapa del SDLC

La información extraída permitió identificar y clasificar las herramientas de AI aplicadas en cada fase del SDLC. Este análisis refleja cómo dichas herramientas se enfocan en resolver los desafíos específicos de cada etapa, optimizando procesos clave y mejorando la eficiencia general del desarrollo de software.

Planificación: Las herramientas en esta fase apuntan principalmente a optimizar la estimación de recursos y cronogramas mediante tecnologías como el aprendizaje profundo y agentes inteligentes. Estas soluciones destacan por su capacidad para facilitar la planificación automática de tareas, coordinar recursos en pipelines DevOps y generar documentación inicial, proporcionando un soporte predictivo significativo.

Recolección de Requisitos: En esta etapa, las tecnologías de procesamiento de lenguaje natural (NLP) y aprendizaje automático (ML) permiten estructurar la información y transformar el lenguaje natural en especificaciones técnicas. Estas aplicaciones reducen la ambigüedad y los errores en la interpretación de requisitos, mejorando su precisión y claridad para fases posteriores.

Análisis: La etapa de análisis utiliza herramientas como OCR para la extracción de texto, técnicas de NLP para la clasificación de requisitos y ML para la predicción de vulnerabilidades. Estos enfoques garantizan que el análisis se base en datos precisos y confiables, incrementando la calidad y credibilidad en la toma de decisiones.

Diseño: Durante esta fase, las herramientas de AI automatizan la generación de diagramas de flujo, optimizan patrones arquitectónicos y permiten la transformación de bocetos en código estructurado. Tecnologías como redes neuronales artificiales y algoritmos de optimización facilitan la creación de diseños escalables y robustos.

Desarrollo: La generación automatizada de código y la detección de vulnerabilidades destacan como las principales contribuciones de la AI en esta etapa. Herramientas de AI generativa



como GitHub Copilot optimizan la calidad del software y reducen errores de codificación manual, acelerando significativamente los procesos de desarrollo.

Pruebas: La automatización en pruebas mediante aprendizaje profundo y NLP permite la generación dinámica de casos de prueba y la detección temprana de errores. Estas tecnologías garantizan una mayor cobertura, reducen tiempos de validación y optimizan la confiabilidad del software.

Despliegue: En la fase de despliegue, se emplean sistemas de integración continua y entrega continua (CI/CD) impulsados por AI para garantizar una liberación eficiente y estable del software. Además, los sistemas predictivos y auto-adaptativos optimizan la gestión de recursos y monitorean el rendimiento en tiempo real, detectando fallos de manera proactiva.

Mantenimiento: Finalmente, la etapa de mantenimiento se fortalece con herramientas de aprendizaje automático que analizan deuda técnica y predicen métricas de mantenibilidad. La reparación automática de errores y el monitoreo predictivo aseguran la estabilidad operativa y prolongan la vida útil del software.

En conjunto, esta clasificación destaca cómo las herramientas de AI asumen roles específicos y diferenciados en cada etapa del SDLC. Sin embargo, su implementación también acarrea desafíos significativos que requieren soluciones innovadoras y adaptativas. Este análisis no solo ofrece un panorama claro de las aplicaciones de la IA, sino que también establece las bases para un examen comparativo que explore tanto los beneficios como las limitaciones de estas tecnologías.

4.3 Desafíos, Factores Contribuyentes, Estrategias Sugeridas e Impacto en el SDLC

El Anexo 2 detalla la información extraída de la literatura, organizada en función de las etapas del SDLC, los desafíos identificados, los factores contribuyentes, estrategias sugeridas y el impacto en la eficiencia y calidad del software. A continuación se presenta una síntesis de la misma.

La matriz de extracción de datos, muestra de forma sistemática cómo se manifiestan los desafíos en las diferentes fases del SDLC en la integración de la IA. Al analizar la información extraída, se pueden identificar patrones y tendencias que evidencian tanto las dificultades principales como las estrategias sugeridas para superarlas.



4.3.1 Temáticas y Desafíos Recurrentes

- **Planificación:** En esta fase, se destacan dos grandes desafíos: la complejidad para estimar el esfuerzo del proyecto y la integración de diversas herramientas y tecnologías. La falta de datos precisos y de modelos predictivos robustos (Durrani et al., 2024; Benitez & Serrano, 2023) se repite como una constante, lo que obliga a los investigadores a proponer la utilización de modelos de ML y deep learning con el fin de mejorar la precisión en la asignación de recursos y tiempos. Asimismo, la complejidad en la integración, debido a la carencia de estándares uniformes, es abordada mediante la recomendación de herramientas interoperables y plataformas estandarizadas (Kambala, 2024).
- **Diseño:** La fase de diseño presenta como desafío la dificultad para automatizar el proceso arquitectónico y la identificación tardía de errores. Se menciona la limitación de las herramientas tradicionales para generar diseños óptimos, lo que ha motivado la propuesta de implementar técnicas de NLP para la generación automática de diagramas y el uso de redes neuronales para la predicción de fallos, lo que reduce la incidencia de errores y agiliza el proceso (Benitez & Serrano, 2023; Garg, K., 2023).
- **Desarrollo:** En la etapa de desarrollo se subraya la importancia de la detección temprana de problemas de seguridad. La dependencia de la revisión manual y la insuficiencia de herramientas eficaces para identificar vulnerabilidades son puntos recurrentes. La solución sugerida consiste en la aplicación de algoritmos de ML para la detección proactiva de fallas, de manera que se reduzca la cantidad de errores en el código (Durrani et al., 2024; Khan, Fatima & Ahmed, 2023).
- **Pruebas:** La matriz muestra que la cobertura insuficiente de casos de prueba y el uso de métodos tradicionales limitan la efectividad del aseguramiento de calidad. Como respuesta, se propone la automatización de la generación y clasificación de casos de prueba utilizando aprendizaje profundo y técnicas NLP, lo que permite aumentar la cobertura de pruebas y reducir el tiempo invertido en la validación (Durrani et al., 2024; Benitez & Serrano, 2023).
- **Despliegue y Mantenimiento:** En estas fases se encuentran desafíos relacionados con la predicción y prevención de defectos, así como con la gestión de la escalabilidad y la



integración con sistemas heredados. La matriz resalta que la utilización de herramientas basadas en AI explicable y modelos predictivos avanzados mejora la confiabilidad del mantenimiento y la estabilidad del sistema, optimizando el rendimiento en entornos dinámicos (Durrani et al., 2024; Kambala, 2024).

4.3.2 Estrategias y Recomendaciones Identificadas

La síntesis de la información extraída revela un consenso en la literatura respecto a las soluciones planteadas para superar los desafíos identificados:

- **Uso de Modelos de Aprendizaje Automático y Profundo:** Se propone recurrentemente la implementación de algoritmos de ML y deep learning para abordar la falta de datos precisos y para mejorar la estimación, la detección de vulnerabilidades y la predicción de mantenimiento (Benitez & Serrano, 2023; Durrani et al., 2024).
- **Automatización y Herramientas Generativas:** En fases como diseño y desarrollo, la automatización de tareas mediante técnicas basadas en NLP y agentes inteligentes se presenta como una estrategia efectiva para acelerar el proceso y reducir errores (Garg, K., 2023; Keskar, 2023).
- **Capacitación y Definición de Estándares:** Varias referencias señalan la importancia de la capacitación continua de los equipos de desarrollo y la adopción de estándares y plataformas interoperables para facilitar la integración de herramientas de AI en entornos tradicionales (Kambala, 2024; Alenezi & Akour, 2025).
- **Implementación de Sistemas Predictivos y de Monitoreo:** Para las fases de despliegue y mantenimiento, se destacan estrategias que involucren la creación de sistemas predictivos basados en AI explicable, lo que permite detectar anomalías y prevenir fallos antes de que impacten al usuario final (Müller, 2024; Kambala, 2024).

4.3.3 Detección de Tendencias y Vacíos de la Literatura

La revisión de la matriz permite observar que, aunque existe un consenso en la utilización de técnicas avanzadas para mejorar la eficiencia en cada fase, también se evidencia una fragmentación en la investigación respecto a la integración simultánea de aspectos técnicos, organizativos y éticos. En particular, mientras se abordan con detalle los desafíos técnicos



(calidad de datos, algoritmos de detección y automatización de tareas), la literatura resulta menos homogénea en la propuesta de soluciones integrales que combinen también la dimensión organizacional y ética para facilitar la adopción de la IA. Este vacío sugiere la necesidad de estudios que articulen un marco holístico capaz de integrar estos tres aspectos, abriendo una línea de investigación para el desarrollo de estrategias completas y adaptativas.

En resumen, la matriz de extracción de datos no solo permite identificar patrones comunes, como la dependencia de datos precisos, la falta de compatibilidad con sistemas heredados y la resistencia organizacional, sino que también destaca las estrategias predominantes para mitigarlos. Estas estrategias, que incluyen desde el uso de modelos de aprendizaje profundo y técnicas de NLP hasta la implementación de sistemas predictivos y programas de capacitación, conforman la base sobre la cual se sustentará el análisis detallado en las secciones posteriores del trabajo. La integración de estos hallazgos fortalece la argumentación de que, a pesar del potencial transformador de la AI en el SDLC, la adopción plena y exitosa requiere una consideración simultánea de los factores técnicos, organizacionales y éticos.

4.3.4 Identificación de las Etapas Críticas del SDLC en la Integración de IA

El análisis de la matriz de extracción de datos revela que, aunque la integración de la AI tiene aplicaciones en todas las fases del SDLC, existen etapas en las cuales los desafíos y barreras son particularmente críticos. A continuación, se destacan tres fases que, según la información recopilada, presentan los mayores obstáculos:

1. Planificación

La fase de planificación se muestra como una de las etapas más desafiantes para la integración de la IA. En la matriz se identifican problemas relacionados con la estimación del esfuerzo del proyecto y la complejidad en la integración de herramientas, señalados en diferentes estudios (Durrani et al., 2024; Benitez & Serrano, 2023; Khan, Fatima & Ahmed, 2023). Entre los factores contribuyentes se encuentran la falta de datos precisos y la ausencia de modelos predictivos sólidos, lo cual dificulta la optimización de recursos y la asignación adecuada de tiempos. Las estrategias sugeridas incluyen el uso de modelos de aprendizaje profundo para mejorar las estimaciones y la adopción de plataformas interoperables que faciliten la integración con sistemas heredados, aspectos resaltados en estudios como los de Kambala (2024).



2. Pruebas y Garantía de Calidad

La fase de pruebas es otra etapa crítica en donde los desafíos se vuelven significativos. La matriz evidencia que la cobertura insuficiente de casos de prueba, la dependencia de métodos tradicionales y la complejidad inherente a la automatización son barreras recurrentes (Durrani et al., 2024; Benitez & Serrano, 2023; Khan, Fatima & Ahmed, 2023). Los factores que dificultan la implementación de soluciones basadas en AI en esta fase incluyen la escasez de herramientas capaces de generar y clasificar dinámicamente casos de prueba, lo que a menudo se traduce en una validación menos robusta del software. Para contrarrestar estos problemas, se sugieren estrategias como el empleo de técnicas de aprendizaje profundo y NLP para automatizar la generación de escenarios de prueba, lo que ha mostrado aumentar tanto la cobertura como la precisión en la validación del software.

3. Despliegue y Mantenimiento

Las fases de despliegue y mantenimiento también se destacan por los desafíos técnicos y organizativos que afectan la integración de la IA. Según la matriz, la integración con sistemas heredados y la escalabilidad en entornos en constante cambio constituyen barreras importantes (Durrani et al., 2024; Kambala, 2024; Vadde & Munagandla, 2022). La dificultad para predecir y prevenir defectos, así como la falta de automatización en el mantenimiento, se presentan como retos relevantes. Estrategias basadas en sistemas predictivos, como el uso de algoritmos de aprendizaje reforzado y de métodos de monitoreo proactivo, se proponen para mejorar la estabilidad y sostenibilidad de los sistemas, lo que impacta directamente en la calidad y confiabilidad del software en producción.

En síntesis, la matriz de datos permite identificar que las fases de **Planificación, Pruebas y Garantía de Calidad, y Despliegue y Mantenimiento** son especialmente críticas y presentan los mayores desafíos y barreras para la integración de la AI en el SDLC. Estos hallazgos subrayan la necesidad de desarrollar modelos predictivos más robustos, automatizar de manera eficaz los procesos de validación y establecer estrategias que integren de forma organizada la tecnología de AI con los sistemas tradicionales.

4.3.5 Exploración de los Factores Técnicos, Organizacionales y Éticos que Dificultan la Adopción de AI en el SDLC



La adopción de la IA en el SDLC es un proceso complejo que se ve obstaculizado por diversos factores. A partir de la sistematización de la información extraída de los estudios analizados, se identifican tres grandes grupos de factores: técnicos, organizacionales y éticos. Cada uno de estos conjuntos de factores incide de manera particular en las etapas del SDLC, afectando la implementación y el impacto de las herramientas de IA. A continuación, se detalla cada categoría:

1. Factores Técnicos

Los desafíos técnicos forman la base de muchas de las barreras para la integración de la IA. Entre los factores más destacados se encuentran:

- **Calidad de Datos y Modelos Predictivos:** Varios estudios coinciden en que la precisión de los modelos de AI depende crucialmente de la disponibilidad de datos limpios y estructurados. La falta de datos precisos y modelos predictivos robustos afecta directamente la fase de planificación, dificultando la estimación de esfuerzo y recursos (Durrani et al., 2024; Benitez & Serrano, 2023). Asimismo, en etapas como el análisis y la validación, la presencia de datos fragmentados y la naturaleza “caja negra” de los algoritmos limitan la capacidad de los sistemas para generar resultados confiables (Özpolat et al., 2023; Siddique, 2022).
- **Integración con Sistemas Existentes:** La incompatibilidad entre las herramientas modernas de AI y las arquitecturas heredadas representa otro desafío técnico importante. En la fase de planificación y despliegue, la falta de estándares universales y el difícil acoplamiento con sistemas tradicionales requieren el uso de plataformas interoperables y soluciones modulares (Kambala, 2024; Haghsheno, 2020).
- **Automatización y Generación de Código:** Durante el desarrollo, el uso de asistentes de código inteligente, como GitHub Copilot y otros generadores automáticos, ha sido una solución prometedora. Sin embargo, la dependencia excesiva de estos sistemas y la necesidad de grandes volúmenes de datos para su entrenamiento pueden introducir errores o vulnerabilidades, afectando la calidad final del código (Khan, Fatima & Ahmed, 2023; Upadhyaya, 2022). En consecuencia, se hace necesario combinar la automatización con revisiones humanas que aseguren la corrección y robustez del código.



2. Factores Organizacionales

Además de las limitaciones técnicas, existen barreras de carácter organizacional que dificultan la adopción efectiva de la IA:

- **Resistencia al Cambio y Falta de Capacitación:** Varios estudios resaltan que la introducción de tecnologías de AI genera incertidumbre en los equipos de desarrollo, principalmente por el temor a la pérdida de roles o a la disminución de la relevancia de habilidades tradicionales (Haghsheno, 2020; Alenezi & Akour, 2025). La carencia de programas de capacitación y actualización profesional genera una brecha que impide que el personal se adapte adecuadamente a las nuevas herramientas.
- **Soporte Gerencial y Necesidad Estratégica:** La limitada percepción de necesidad organizacional y el apoyo insuficiente por parte de la alta dirección también son factores que frenan la integración de la IA. La adopción de tecnologías innovadoras requiere una visión estratégica clara y recursos asignados específicamente para su implementación, aspecto que no siempre está presente en los entornos tradicionales (Giffari et al., 2024; Ricárdez et al., 2024).

3. Factores Éticos

Finalmente, los aspectos éticos juegan un papel fundamental en la adopción de la IA, particularmente en lo que respecta a la confianza y la equidad:

- **Privacidad y Seguridad de los Datos:** La preocupación por la privacidad es recurrente en diversas fases, especialmente cuando se trata de manejar grandes volúmenes de información sensible. La exposición de datos confidenciales y la gestión inadecuada de la seguridad pueden comprometer tanto la fiabilidad de los sistemas como la confianza de los usuarios (Todorov, 2022; Russo, 2024).
- **Transparencia y Explicabilidad:** La falta de explicabilidad en muchos modelos de IA, que funcionan de manera similar a una “caja negra”, dificulta la comprensión de las decisiones automatizadas. Este problema ético impide que los desarrolladores y stakeholders confíen plenamente en las herramientas, limitando su adopción y su integración en procesos críticos (Todorov, 2022; Russo, 2024).



La revisión de la información extraída revela que, si bien la integración de la AI ofrece numerosas ventajas en el desarrollo de software, su adopción está significativamente condicionada por desafíos técnicos, organizacionales y éticos. Los problemas técnicos relacionados con la calidad de los datos y la integración con sistemas heredados, las barreras organizacionales como la resistencia al cambio y la falta de capacitación, y las preocupaciones éticas sobre la privacidad y la transparencia, conforman un entramado complejo que limita la plena aplicación de soluciones basadas en AI en el SDLC.

4.3.6 Ejemplos Prácticos y Estudios de Caso en la Integración de AI en el SDLC

La revisión de la literatura y la sistematización de los datos extraídos han permitido identificar numerosos estudios de caso y ejemplos prácticos que evidencian los desafíos recurrentes en la integración de la AI a lo largo del ciclo de vida del desarrollo de software (SDLC). A continuación se destacan algunos de los casos más representativos:

1. **Automatización en DevOps y Despliegue:** Estudios como los de Kambala (2024) y Durrani et al. (2024) presentan ejemplos concretos de la implementación de agentes inteligentes para automatizar pipelines de integración y despliegue continuo (CI/CD). Estos casos demuestran cómo la adopción de herramientas de AI puede mejorar la coordinación entre equipos y optimizar la entrega de software. Sin embargo, también resaltan desafíos, como la dificultad en la integración con sistemas heredados y la necesidad de ajustar procesos internos para adaptarse a nuevas tecnologías.
2. **Recolección y Validación de Requisitos Mediante NLP:** Autores como Garg, K. (2023) y Upadhyaya (2022) han documentado casos en los que se han utilizado técnicas de NLP para extraer y estructurar información de documentos y retroalimentación de usuarios. Estos estudios evidencian cómo la automatización en la recolección de requisitos permite disminuir la ambigüedad y mejorar la calidad de los datos, lo que a su vez reduce errores en fases posteriores del SDLC.
3. **Automatización y Validación en la Fase de Pruebas:** Los estudios de Benitez & Serrano (2023) y Khan, Fatima & Ahmed (2023) incluyen ejemplos prácticos en los que la integración de técnicas de aprendizaje profundo y NLP ha permitido automatizar la generación y clasificación de casos de prueba. Estos casos prácticos subrayan la importancia de aumentar la cobertura y cumplir con los estándares de calidad, a la vez



que evidencian la reducción del tiempo y esfuerzo manual en la validación del software.

4. **Mantenimiento Predictivo y Soporte:** La literatura, a través de trabajos como los de Müller (2024) y Ajiga et al. (2024), ofrece ejemplos de cómo los sistemas predictivos basados en AI han sido implementados para monitorear y mantener la estabilidad de aplicaciones en entornos de producción. Estos estudios muestran que, mediante la detección proactiva de defectos y la aplicación de algoritmos de mantenimiento predictivo, es posible reducir sustancialmente los tiempos de inactividad y mejorar la confiabilidad del software.

Estos estudios de caso y ejemplos prácticos no solo evidencian los desafíos recurrentes, como la dependencia de datos de alta calidad, la compatibilidad con sistemas existentes y la necesidad de capacitación constante, sino que también demuestran la aplicabilidad de diferentes estrategias y herramientas de AI para superarlos. La diversidad y profundidad de estos ejemplos han permitido identificar patrones comunes que refuerzan la importancia de abordar estos desafíos de manera integral, combinando soluciones técnicas con cambios organizacionales y políticas éticas.

4.4 Análisis de la Calidad de los Artículos Seleccionados

En esta investigación se evaluaron los artículos seleccionados mediante una matriz de evaluación (anexo 3), que consideró cinco criterios fundamentales: claridad en los objetivos, metodología empleada, reproducibilidad, validez de los resultados y relevancia de las contribuciones. A continuación, se presenta un análisis detallado de la calidad y características clave de los estudios revisados.

1. Claridad en los Objetivos

- **Fortalezas Identificadas:** La mayoría de los artículos evaluados definieron claramente sus objetivos, alineándolos directamente con el tema de investigación, la integración de la AI en el SDLC. Por ejemplo, trabajos como los de Russo (2024) y Durrani et al. (2024) destacan por formular objetivos concretos y bien justificados que abordan tanto desafíos como oportunidades. Estos estudios establecen un marco claro para entender la importancia de la AI en diferentes fases del SDLC.



- **Limitaciones Detectadas:** Algunos artículos, aunque relevantes, presentaron una falta de especificidad en sus objetivos, limitándose a abordar problemas generales sin profundizar en casos prácticos específicos. Esto se observa, por ejemplo, en Ricárdez et al. (2024), donde los objetivos podrían enfocarse más en los desafíos técnicos particulares de ciertas etapas del SDLC.

2. Metodología Utilizada

- **Fortalezas Identificadas:** Varios estudios emplearon enfoques metodológicos robustos y bien documentados. El trabajo de Russo (2024), por ejemplo, combina métodos cualitativos y cuantitativos utilizando validación estadística, lo que refuerza la confiabilidad de sus hallazgos. Asimismo, Durrani et al. (2024) sigue una metodología sistemática basada en el enfoque de Kitchenham, ampliamente reconocido en el ámbito de la ingeniería de software.
- **Limitaciones Detectadas:** La reproducibilidad metodológica fue un área de mejora en muchos estudios. Aunque algunos describen las herramientas y técnicas utilizadas, como Kambala (2024) y Benitez & Serrano (2023), a menudo carecen de detalles técnicos necesarios para replicar completamente los experimentos o análisis.

3. Reproducibilidad

- **Fortalezas Identificadas:** Estudios como el de Durrani et al. (2024) destacaron por proporcionar descripciones detalladas de sus procedimientos, facilitando la replicación futura de su trabajo. Esto asegura un nivel alto de trazabilidad y rigor científico.
- **Limitaciones Detectadas:** Otros estudios, como los de Garg (2023) y Medepalli (2025), aunque presentan ejemplos interesantes, no incluyen suficiente información técnica para garantizar una replicabilidad completa. Esto limita su utilidad como referencia en investigaciones similares.

4. Validez de los Resultados

- **Fortalezas Identificadas:** La mayoría de los artículos respaldaron sus conclusiones con análisis sólidos y basados en evidencia empírica o estudios previos relevantes. Por ejemplo, trabajos como el de Russo (2024) se apoyaron en métodos estadísticos robustos y un tamaño de muestra significativo para validar sus modelos teóricos.

- **Limitaciones Detectadas:** Algunos artículos tienden a generalizar sus hallazgos sin ofrecer suficiente evidencia cuantitativa o experimental que respalde las conclusiones, como Siddique (2022). Este enfoque puede limitar la aplicabilidad de sus resultados a contextos más amplios.

5. Relevancia y Contribución

- **Fortalezas Identificadas:** Todos los artículos incluidos en esta revisión abordaron temas altamente pertinentes, enfocándose en áreas clave como la automatización, la AI generativa y los desafíos éticos en el desarrollo de software. Trabajos como los de Vadde & Munagandla (2022) y Upadhyaya (2022) se destacan por su aplicabilidad práctica en entornos actuales de ingeniería de software.
- **Limitaciones Detectadas:** Aunque los artículos son relevantes, algunos repiten hallazgos previos sin proponer innovaciones significativas. Esto se observa en estudios como el de Medepalli (2025), que reitera puntos generales ya conocidos en la literatura.

Patrones y Tendencias Generales

El análisis de los artículos seleccionados permite identificar varias tendencias relevantes:

1. **Diversidad de enfoques metodológicos:** Se observa un equilibrio entre estudios descriptivos, revisiones sistemáticas y análisis cualitativos. Esto enriquece el panorama de la investigación, pero también refleja una necesidad de mayor estandarización en la metodología.
2. **Foco en fases específicas del SDLC:** La mayoría de los artículos se concentran en etapas como DevOps, pruebas y desarrollo, mientras que fases como la planificación y el mantenimiento reciben menos atención.
3. **Impacto creciente de la AI generativa:** Herramientas como ChatGPT, Codex, GitHub Copilot son recurrentemente discutidas en los estudios, destacando su potencial para transformar el SDLC y automatizar tareas clave.

La calidad general de los artículos seleccionados es alta, con un énfasis significativo en los temas más relevantes para la integración de la AI en el SDLC. Sin embargo, se identificaron



limitaciones relacionadas con la reproducibilidad metodológica y la profundidad en la validación empírica de los resultados. Estas observaciones subrayan la importancia de seguir fortaleciendo las prácticas de investigación en este campo para garantizar resultados más sólidos y generalizables.

Capítulo 5: ANALISIS Y DISCUSIÓN

La integración de la AI en el SDLC ha sido ampliamente estudiada, y la evidencia recopilada en esta revisión narrativa muestra un panorama complejo y, en ocasiones, contradictorio. Si bien múltiples estudios destacan el potencial transformador de la IA, también se evidencia una serie de desafíos que no solo se limitan a aspectos técnicos, sino que se entrelazan con barreras organizativas y consideraciones éticas. En esta discusión se abordarán cada uno de estos motivos, con una mirada crítica que reconoce tanto los avances como las limitaciones y controversias derivadas de la implementación de la AI en entornos reales.

5.1 Desafíos Técnicos: Entre el Potencial y las Limitaciones

Aunque es indudable que el uso de modelos de aprendizaje profundo y técnicas de NLP ha permitido avances en áreas como la planificación y la automatización de pruebas (Benitez & Serrano, 2023; Durrani et al., 2024), los datos extraídos revelan que la precisión y la fiabilidad de estos sistemas aún dependen en gran medida de la calidad de los datos.

- **Calidad de Datos y Modelos Predictivos:** En teoría, la utilización de algoritmos de ML para predecir el esfuerzo en la planificación y detectar vulnerabilidades en el desarrollo sugiere un gran avance. No obstante, la falta de datos precisos y la persistente “caja negra” en muchos modelos generan incertidumbre. Esto pone en entredicho la escalabilidad de estas soluciones en entornos con datos fragmentados o en constante cambio. Aquí surge una controversia: ¿realmente se puede confiar ciegamente en estos modelos o es necesaria una revisión continua que combine la automatización con una supervisión humana intensa? Se plantea que, si bien los avances son prometedores, los modelos actuales aún tienen un largo camino para lograr la robustez exigida en entornos críticos.



- **Integración con Sistemas Existentes:** Las estrategias técnicas propuestas, como la adopción de plataformas interoperables y arquitecturas modulares, son frecuentemente recomendadas (Kambala, 2024; Haghsheno, 2020). Sin embargo, la realidad es que las empresas a menudo implementan soluciones puntuales que no logran integrarse de forma holística con sus sistemas heredados. Esto sugiere una brecha significativa entre la teoría y la práctica: por un lado, los enfoques modulares son conceptualmente atractivos, pero la transición en sistemas reales, que han evolucionado durante años sin considerar la IA, sigue siendo un reto considerable y controversial.

5.2 Barreras Organizacionales y Transformación Cultural

El factor humano y organizacional se destaca como uno de los más críticos y también de mayor controversia en la literatura.

- **Resistencia al Cambio y Falta de Capacitación:** Los estudios señalan que el temor al desplazamiento laboral y la pérdida de competencias tradicionales generan resistencia frente a la adopción de AI (Haghsheno, 2020; Alenezi & Akour, 2025). En este sentido, aunque muchos autores recomiendan programas de capacitación y actualización, en la práctica, la inversión en formación varía notablemente entre organizaciones. Esto refleja una disparidad: mientras algunos líderes tecnológicos ven en la AI un aliado indispensable, otros se muestran escépticos ante el costo y la curva de aprendizaje asociados. Esta división crea un terreno controversial donde la innovación tecnológica choca con la realidad organizacional y cultural.
- **Soporte Gerencial y Estrategia de Innovación:** La falta de un compromiso estratégico claro y la limitada asignación de recursos impiden la creación de un entorno favorable para la integración de la IA. Este aspecto es particularmente polémico, ya que algunas organizaciones han logrado implementaciones exitosas mediante la creación de equipos interdisciplinarios y un fuerte soporte gerencial, mientras que en otras, la adopción se ha visto obstaculizada por una cultura organizacional poco propensa al cambio. La controversia reside en determinar si la resistencia es inherente a la naturaleza humana o si se trata, además, de una falla en la comunicación y en la estrategia de liderazgo.



5.3 Consideraciones Éticas y el Desafío de la Transparencia

Los aspectos éticos surgen como otra dimensión crítica y a menudo polémica en la integración de la IA.

- **Privacidad y Seguridad de los Datos:** La preocupación por la privacidad es recurrente en múltiples estudios (Todorov, 2022; Russo, 2024). La utilización de grandes volúmenes de información sensible expone a las organizaciones a riesgos significativos, y, en este campo, la falta de normativas claras y la ausencia de auditorías rigurosas generan debates acalorados acerca de hasta qué punto es seguro implementar la automatización basada en AI sin comprometer la confidencialidad de los datos.
- **Transparencia y Explicabilidad:** La transparencia de los modelos de AI es otro punto de crítica. La dificultad para interpretar las decisiones automatizadas genera desconfianza entre los usuarios y desarrolla una percepción negativa respecto a la "inteligencia" real de estos sistemas. Aunque se proponen soluciones como la implementación de modelos explicables, muchos expertos consideran que estas técnicas todavía son incipientes y no ofrecen la claridad necesaria para cumplir con estándares regulatorios y éticos, lo que plantea un área controvertida que requiere más investigación y consenso.

5.4 Síntesis de la Controversia y Proyecciones Futuras

Si bien la literatura ofrece numerosas estrategias para mitigar estos desafíos, como el uso de sistemas predictivos, la adopción de arquitecturas modulares, la implementación de programas de capacitación y la generación de modelos explicables, la implementación práctica de estas soluciones aún presenta dificultades. La disparidad entre la teoría y la práctica se muestra en la diversidad de enfoques y la falta de consenso en la comunidad sobre las mejores prácticas. Esto sugiere que, para lograr una integración exitosa de la AI en el SDLC, es necesaria una visión multidimensional que combine los aspectos técnicos con una transformación cultural y el establecimiento de marcos éticos sólidos.

En conclusión, la integración de la AI en el desarrollo de software enfrenta desafíos complejos y multidimensionales, que requieren una respuesta integrada. La evidencia analizada destaca que, a pesar de las mejoras introducidas por modelos avanzados y herramientas automatizadas, la problemática se extiende a aspectos organizativos y éticos que continúan



siendo fuente de controversia. Este escenario pone de manifiesto la necesidad de adoptar estrategias que no solo se enfoquen en superar barreras técnicas, sino que también fomenten un cambio cultural y la implementación de políticas éticas claras. La convergencia de estos elementos es esencial para maximizar el potencial transformador de la AI en el SDLC y sentar las bases para futuras investigaciones que aborden los retos desde una perspectiva holística.

5.5 Recomendaciones

Propuesta de Estrategias para Superar los Desafíos en la Integración de AI en el SDLC

A partir del análisis de la matriz de extracción de datos y la revisión de la literatura, se han identificado barreras recurrentes en diversas fases del SDLC para la integración de la IA. Para promover una adopción exitosa de la IA, se proponen estrategias fundamentadas que aborden de forma integrada los desafíos técnicos, organizativos y éticos. A continuación, se presentan las recomendaciones clave:

1. Estrategias Técnicas

- **Mejora de la calidad de los datos y validación de modelos:**
 - *Proceso de Curación de Datos:* Implementar procesos de preprocesamiento y curación de datos que aseguren la calidad y la integridad de la información utilizada para entrenar modelos predictivos. Estos procesos deben incluir técnicas de limpieza, validación y estructuración que ayuden a mitigar la fragmentación de los datos (Benitez & Serrano, 2023; Durrani et al., 2024).
 - *Modelos de AI Explicables:* Desarrollar o incorporar técnicas de AI explicable (por ejemplo, mapas de atención y modelos basados en reglas) para reducir la opacidad de los modelos y facilitar la interpretación de sus decisiones, aumentando así la confianza tanto de los desarrolladores como de los usuarios (Özpolat et al., 2023; Todorov, 2022).
- **Integración con sistemas heredados y estandarización:**
 - *Plataformas Interoperables:* Adoptar herramientas y plataformas estandarizadas que aseguren la interoperabilidad entre los sistemas



tradicionales y las soluciones basadas en IA, lo que facilitará la integración y reducirá los costos de adaptación (Kambala, 2024).

- *Sistemas Modulares y Auto-Adaptativos:* Diseñar arquitecturas modulares que permitan la integración gradual de la IA, adaptándose a las particularidades de cada entorno y posibilitando escalabilidad dinámica a través de algoritmos de aprendizaje reforzado (Müller, 2024; Vadde & Munagandla, 2022).

2. Estrategias Organizacionales

- **Capacitación y Desarrollo de Competencias:**

- *Programas de Formación Continua:* Implementar programas de capacitación y actualización dirigidos a los equipos de desarrollo, orientados a fortalecer tanto el conocimiento técnico en AI como el manejo de herramientas automatizadas. Esto ayudará a superar la resistencia al cambio y a reducir la brecha de habilidades (Haghsheno, 2020; Alenezi & Akour, 2025).
- *Equipos Interdisciplinarios:* Promover la formación de equipos que integren especialistas en IA, desarrolladores de software y gestores de proyectos. La sinergia entre estos perfiles facilitará la adopción de soluciones híbridas y la implementación de prácticas innovadoras.

- **Apoyo Gerencial y Estrategia de Innovación:**

- *Compromiso de la Alta Dirección:* Sensibilizar a la alta dirección sobre la importancia estratégica de la IA, respaldando la asignación de recursos y la definición de políticas de innovación que favorezcan la adopción de tecnologías emergentes.
- *Cultura Organizacional Abierta:* Fomentar una cultura de aprendizaje y adaptabilidad que valore el uso de nuevas tecnologías y promueva la experimentación controlada, reduciendo la resistencia organizacional al cambio.

3. Estrategias Éticas



- **Marco de Ética y Gobernanza de Datos:**

- *Auditorías y Políticas de Privacidad:* Establecer un marco regulatorio que incluya políticas de privacidad y seguridad de datos, junto con auditorías periódicas para detectar y mitigar riesgos éticos y de sesgos en los modelos de AI (Todorov, 2022; Russo, 2024).
- *Transparencia en la Toma de Decisiones:* Implementar mecanismos que aseguren la explicabilidad de los resultados generados por la IA, fortaleciendo la confianza de todos los stakeholders y garantizando el cumplimiento de normativas éticas.

5.6 Limitaciones del Estudio

Este estudio, al ser una revisión narrativa, presenta ciertas limitaciones inherentes al enfoque metodológico empleado. Una de las principales limitaciones radica en el sesgo introducido por la selección de los artículos incluidos en la revisión. La narrativa se construye en función de las decisiones del autor sobre qué estudios y publicaciones incorporar, lo que puede influir en los resultados y las conclusiones obtenidas. Además, la revisión narrativa no siempre sigue un protocolo sistemático para la búsqueda y selección de la literatura, lo que podría resultar en la omisión involuntaria de estudios relevantes. Esto limita la capacidad de generalizar las conclusiones y podría impactar la validez externa del estudio.

En futuros estudios, sería beneficioso complementar este enfoque con revisiones sistemáticas que empleen criterios de inclusión y exclusión claros, así como técnicas reproducibles para la búsqueda y el análisis de la literatura. Esto permitiría reducir el sesgo inherente y garantizar un panorama más exhaustivo y equilibrado de la temática estudiada.

Capítulo 6: CONCLUSIÓN

Esta revisión narrativa titulada “Desde los Obstáculos hasta las Estrategias: Integración de la inteligencia artificial en el ciclo de vida del desarrollo del software” ha permitido analizar de forma exhaustiva los retos y obstáculos que impiden una adopción plena y efectiva de la AI en el SDLC. La sistematización de la información, a través de la matriz de extracción de datos y la clasificación de herramientas de AI por etapas, ha revelado que, a pesar de los avances



notables en áreas como la automatización y la generación de código, siguen persistiendo desafíos críticos.

En el ámbito **técnico**, se evidencia que la precisión y robustez de los modelos dependen fuertemente de la calidad de los datos utilizados. La ausencia de datos consistentes y los problemas de “caja negra” en muchos algoritmos generan incertidumbre sobre la validez de los resultados, lo que limita su aplicación en entornos reales. Por ello, es imperativo mejorar los procesos de curación y validación de datos para fortalecer los modelos predictivos.

Desde la perspectiva **organizacional**, la resistencia al cambio y la falta de capacitación constituyen barreras significativas. Mientras algunas organizaciones logran incorporar con éxito la AI mediante el fortalecimiento de equipos interdisciplinarios y un sólido soporte gerencial, otras enfrentan dificultades debido a la limitada inversión en formación técnica y a la falta de una cultura de innovación. Esta disparidad destaca la importancia de desarrollar estrategias que promuevan la adaptación cultural y el compromiso de todos los niveles de la organización.

En cuanto a los **aspectos éticos**, la privacidad, seguridad y transparencia son temas recurrentes. La preocupación por el manejo de información sensible y la falta de mecanismos claros que expliquen el funcionamiento de los modelos de AI dificultan la confianza de los usuarios y desarrolladores. Es esencial que las organizaciones implementen marcos regulatorios y políticas de gobernanza que garanticen una adopción ética y responsable de estas tecnologías.

La evidencia recopilada, junto con los estudios de caso presentados, sugiere que las áreas de planificación, pruebas y despliegue/ mantenimiento son especialmente críticas. La aplicación exitosa de la AI en estas fases depende de la capacidad para integrar soluciones técnicas innovadoras con un sólido soporte organizacional y marcos éticos claros.

En resumen, aunque la integración de la AI ofrece la posibilidad de transformar radicalmente los procesos de desarrollo de software, su implementación efectiva exige superar desafíos que son, a su vez, técnicos, organizativos y éticos. Las estrategias recomendadas, que abarcan la mejora de la calidad de los datos, el diseño de arquitecturas modulares, la capacitación continua y el establecimiento de normativas éticas, forman la base para una adopción más robusta y sostenible de la AI en el SDLC.



UNIVERSIDAD
NACIONAL
DE LA PLATA



**“Desde los Obstáculos hasta las Estrategias: Integración
de la inteligencia artificial en el ciclo de vida del
desarrollo del software.”**

Esta investigación busca ofrecer un enfoque holístico que puede orientar tanto a la academia como a la industria en la transición hacia procesos de desarrollo más inteligentes y adaptables. No obstante, resulta crucial que futuros estudios profundicen en la validación empírica de estas estrategias y desarrollen marcos integrales que aborden de forma simultánea los desafíos técnicos, organizacionales y éticos.



REFERENCIAS BIBLIOGRAFICAS

- Ajiga, D., Okeleke, P. A., Folorunsho, S. O., & Ezeigweneme, C. (2024). Enhancing software development practices with AI insights in high-tech companies. IEEE Software Engineering Institute, Technical Report TR-2024-003.
- Alenezi, M., & Akour, M. (2025). AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions. *Applied Sciences*, 15(3), 1344.
- Alford, G. (2019). Intelligent agents in DevOps: Automating pipelines for enhanced delivery. *Journal of Software Automation*, 12(3), 45-58.
- Al-kfairy, M. (2025). Strategic Integration of Generative AI in Organizational Settings: Applications, Challenges and Adoption Requirements. *IEEE Engineering Management Review*.
- Benitez, C. D., & Serrano, M. (2023). The Integration and Impact of Artificial Intelligence in Software Engineering. *Integration*, 3(2).
- Brown, T., Green, R., & Smith, L. (2018). Organizational barriers in adopting AI-driven software solutions. *International Journal of Business and Technology*, 6(4), 221-230.
- Durrani, U. K., Akpinar, M., Adak, M. F., Kabakus, A. T., Ozturk, M. M., & Saleh, M. (2024). A decade of progress: A systematic literature review on the integration of AI in software engineering phases and activities (2013-2023). *IEEE Access*.
- García, M., López, A., & Sánchez, J. (2022). Ethics in artificial intelligence: Addressing bias and privacy challenges. *European Ethics and Technology Review*, 14(2), 89-102.
- Garg, K. (2023). Impact of Artificial Intelligence on software development: Challenges and Opportunities. *International Journal of Software & Hardware Research in Engineering*, 11(8).
- Giffari, R., Ridho, M. M., Sensuse, D. I., Hidayat, D. S., & Purwaningsih, E. H. (2024). Analyst's Perception on the Use of AI-based Tools in the Software Development Life Cycle. *Jurnal Sistem Informasi*, 20(1), 73-87.
- Haghsheno, S. (2020). Revolutionizing Software Engineering: Leveraging AI for Enhanced Development Lifecycle. *International Journal of Innovative Research in Modern Physics and Software (IJIRMPs)*, 8(1), January-February.
- KAMBALA, G. (2024). Intelligent Software Agents for Continuous Delivery: Leveraging AI and Machine Learning for Fully Automated DevOps Pipelines.



- Keskar, A. (2023). Revolutionizing Software Engineering with Generative AI: Transforming Development Processes, Automation, and Quality Assurance. *TIJER - International Research Journal*, 10(6), 830–844.
- Khan, A. R., Fatima, J., & Ahmed, S. (2023). Crossover of Artificial Intelligence and Software Development Lifecycle. *International Journal of Computing and Related Technologies*, 4(1), 35-42.
- Kokol, P. (2024). The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature. *Information*, 15(6), 354.
- Kumar, S. (2017). Hybrid models in artificial intelligence: Supervising automation effectively. *Software Engineering Perspectives*, 8(1), 17-29.
- Kumar, S. (2024). Artificial Intelligence in Software Engineering: A Systematic Exploration of AI-Driven Development.
- Lee, Y., & Park, H. (2015). Adaptive frameworks for software integration: A theoretical approach. *Engineering Studies Quarterly*, 9(3), 112-128.
- Madupati, B. (2025). AI's Impact on Traditional Software Development. *arXiv preprint arXiv:2502.18476*.
- Medepalli, S. (2025). AI In Modern Software Development: Current Practices, Challenges and Future Possibilities. *Journal Publication of International Research for Engineering and Management (JOIREM)*, 10(01).
- Müller, C. (2020). Transparency challenges in AI models: Overcoming the black-box problem. *Journal of AI Research*, 15(7), 134-145.
- Müller, M. (2024). Advancements in Software Engineering through Artificial Intelligence.
- Oluwaseun, A. W. (2023). Harnessing AI for Advanced Software Engineering: Opportunities and Challenges. *Journal of Emerging Technologies and Innovative Research*, 10(5), 594–599.
- Özpolat, Z., Yıldırım, Ö., & Karabatak, M. (2023). Artificial intelligence-based tools in software development processes: Application of chatgpt. *European Journal of Technique (EJT)*, 13(2), 229-240.
- Pothukuchi, A. S., Kota, L. V., & Mallikarjunaradhya, V. (2023). Impact of generative AI on the software development lifecycle (SDLC). *International Journal of Creative Research Thoughts*, 11(8).



- Ricárdez, J. J. U., Vasconcelos, M. P., Domínguez, R. G., & Romero, F. C. (2024). Integration of Generative AI with ChatGPT in Software Development.
- Russo, D. (2024). Navigating the complexity of generative AI adoption in software engineering. *ACM Transactions on Software Engineering and Methodology*, 33(5), 1-50.
- Sergeyuk, A., Zakharov, I., Koshchenko, E., & Izadi, M. (2025). Human-AI Experience in Integrated Development Environments: A Systematic Literature Review. *arXiv preprint arXiv:2503.06195*.
- Shao, X., Shibasaki, M., & Shibasaki, R. (2025, February). Artificial Intelligence in Software Engineering: Integration and Key Challenges. In *CS & IT Conference Proceedings (Vol. 15, No. 4)*. CS & IT Conference Proceedings.
- Shah, K., & Trehan, A. (2024). Streamlining software development: A comparative study of AI-driven automation tools in modern tech. *International Journal of Computer Engineering and Technology*, 15(6), 1638-1650.
- Siddique, I. (2022). Harnessing artificial intelligence for systems engineering: Promises and pitfalls. *European Journal of Advances in Engineering and Technology*, 9(9), 67-72.
- Simon, J., Taylor, P., & Carter, H. (2021). Reducing errors in software design through deep learning techniques. *Software Development Insights*, 18(5), 99-120.
- Todorov, P. G. (2022). The Application of Artificial Intelligence in Software Engineering. Available at SSRN.
- Upadhyaya, N. (2022). The Role of Artificial Intelligence in Software Development: A Literature Review. ResearchGate.
- Upadhyaya, N. AI-Powered Secure Software Development: The Future of Safe and Efficient Coding.
- Vadde, B. C., & Munagandla, V. B. (2022). AI-Driven Automation in DevOps: Enhancing Continuous Integration and Deployment. *International Journal of Advanced Engineering Technologies and Innovations*, 1(3), 183-193.
- Zhang, Q. (2024). The Role of Artificial Intelligence in Modern Software Engineering. *Applied and Computational Engineering*, 97, 18-23.
- Zhang, Y. (2020). Generative models for collaborative software design. *Journal of Applied AI in Engineering*, 25(6), 321-335.



ANEXOS

Anexo 1: Herramientas de AI en cada etapa del SDLC

En este anexo se expone una clasificación detallada de las herramientas de AI identificadas durante el análisis bibliográfico, los datos extraídos fueron organizados según las etapas del SDLC. Este enfoque permite evidenciar cómo la AI contribuye a optimizar procesos específicos en las distintas fases del desarrollo, desde la planificación hasta el mantenimiento.

Tabla 1. Herramientas de AI Aplicadas en el SDLC.

Etapa	Herramientas IA	Referencia
Planificación	OCR (Reconocimiento Óptico de Caracteres): Usado para extraer texto de imágenes o archivos PDF, facilitando la recolección de requisitos.	Giffari et al. (2024)
Planificación	Modelos de aprendizaje profundo (Deep Learning) para predecir modelos de SDLC y estimar el tamaño del software basado en puntos de función.	Benitez & Serrano (2023)
Planificación	Técnicas de aprendizaje automático (Machine Learning) para especificaciones de requisitos y predicción de vulnerabilidades.	Benitez & Serrano (2023)
Diseño	Uso de Natural Language Processing (NLP) para generar diagramas de flujo de datos (DFD) y modelado impulsado por voz. Redes neuronales artificiales (Artificial Neural Networks, ANN) para predicción de errores en el software.	Benitez & Serrano (2023)
Despliegue	Generación de código mediante AI generativa y clasificación de errores utilizando NLP. Identificación de puntos débiles en el código fuente con técnicas de aprendizaje automático.	Benitez & Serrano (2023)
Pruebas	Automatización de generación de escenarios de prueba con aprendizaje profundo y clasificación de casos de prueba mediante una combinación de NLP y Machine Learning.	Benitez & Serrano (2023)
Mantenimiento	Uso de técnicas de aprendizaje automático para detectar y analizar deuda técnica. Predicción de métricas de mantenibilidad del	Benitez & Serrano (2023)



	software mediante aprendizaje profundo.	
Planificación	Uso de agentes inteligentes para la planificación automática de tareas y asignación de recursos en pipelines DevOps.	Kambala (2024)
Despliegue	Machine Learning para integración continua y detección en tiempo real de errores en los pipelines.	Kambala (2024)
Pruebas	Herramientas basadas en AI para la automatización de pruebas y generación dinámica de casos de prueba.	Kambala (2024)
Despliegue	Agentes inteligentes para la orquestación automatizada y despliegue en entornos de producción.	Kambala (2024)
Planificación	Modelos de aprendizaje profundo (Deep Learning) para predecir modelos de SDLC y optimizar la estimación del tamaño del software.	Durrani et al. (2024)
Planificación	Técnicas de procesamiento de lenguaje natural (Natural Language Processing, NLP) para la clasificación de requisitos y análisis de sentimientos.	Durrani et al. (2024)
Diseño	Algoritmos de optimización para la localización de características y predicción de acciones de reparación de software.	Durrani et al. (2024)
Implementación	Generación de código asistida por AI generativa y detección de vulnerabilidades mediante aprendizaje automático (Machine Learning).	Durrani et al. (2024)
Pruebas	Automatización de generación de casos de prueba con aprendizaje profundo y clasificación de casos de prueba con NLP y ML combinados.	Durrani et al. (2024)
Mantenimiento	Algoritmos de optimización para predecir métricas de mantenibilidad y análisis de deuda técnica con aprendizaje automático.	Durrani et al. (2024)
Planificación	Modelos de aprendizaje profundo (<i>Deep Learning</i>) para predecir modelos de SDLC y estimar el tamaño del software basado en puntos de función.	Khan, Fatima & Ahmed (2023)
Planificación	Técnicas de aprendizaje automático (<i>Machine Learning</i>) para especificaciones de requisitos y predicción de vulnerabilidades.	Khan, Fatima & Ahmed (2023)



Diseño	Uso de <i>Natural Language Processing</i> (NLP) para generar diagramas de flujo de datos (DFD) y modelado impulsado por voz. Redes neuronales artificiales (<i>Artificial Neural Networks, ANN</i>) para predicción de errores en el software.	Khan, Fatima & Ahmed (2023)
Despliegue	Generación de código mediante AI generativa y clasificación de errores utilizando NLP. Identificación de puntos débiles en el código fuente con técnicas de aprendizaje automático.	Khan, Fatima & Ahmed (2023)
Pruebas	Automatización de generación de escenarios de prueba con aprendizaje profundo y clasificación de casos de prueba mediante una combinación de NLP y <i>Machine Learning</i> .	Khan, Fatima & Ahmed (2023)
Mantenimiento	Uso de técnicas de aprendizaje automático para detectar y analizar deuda técnica. Predicción de métricas de mantenibilidad del software mediante aprendizaje profundo.	Khan, Fatima & Ahmed (2023)
Planificación	Uso de técnicas de aprendizaje automático (<i>Machine Learning</i>) y ciencia de datos para analizar y definir el alcance del proyecto.	Garg, K. (2023)
Pruebas	Herramientas basadas en AI que analizan código y datos históricos para identificar y corregir errores de manera eficiente.	Garg, K. (2023)
Codificación	Editores de código inteligentes con funciones de autocompletado basadas en AI para acelerar el desarrollo y reducir errores.	Garg, K. (2023)
Pruebas	Automatización de casos de prueba mediante aprendizaje profundo (<i>Deep Learning</i>) y procesamiento de lenguaje natural (NLP).	Garg, K. (2023)
Mantenimiento	Predicción de métricas de mantenibilidad y análisis de deuda técnica utilizando técnicas de aprendizaje automático.	Garg, K. (2023)
Planificación	Modelos de aprendizaje profundo (<i>Deep Learning</i>) como LSTM y RNN para predicciones de cronogramas y asignación de recursos.	Kumar, S. (2024)
Planificación	Procesamiento de lenguaje natural (<i>Natural Language Processing, NLP</i>) para analizar retroalimentación de usuarios y extraer requisitos clave.	Kumar, S. (2024)



Codificación	Redes neuronales recurrentes (<i>RNNs</i>) y modelos transformadores para generación automática de código y sugerencias de autocompletado.	Kumar, S. (2024)
Pruebas	Aprendizaje no supervisado (<i>Unsupervised Learning</i>) para detección de anomalías y generación inteligente de casos de prueba.	Kumar, S. (2024)
Despliegue	Algoritmos de aprendizaje por refuerzo (<i>Reinforcement Learning</i>) para provisión dinámica de recursos y procesos de auto-recuperación.	Kumar, S. (2024)
Codificación	Modelos de aprendizaje profundo para generación automática de código, como OpenAI Codex y DeepCode.	Müller, M. (2024)
Pruebas	Herramientas de generación automatizada de casos de prueba y localización de defectos mediante aprendizaje automático.	Müller, M. (2024)
Mantenimiento	Sistemas de detección y reparación automática de errores, utilizando técnicas de búsqueda semántica y reparación de programas.	Müller, M. (2024)
Despliegue	Sistemas auto-adaptativos basados en aprendizaje por refuerzo para optimizar recursos y garantizar la estabilidad.	Müller, M. (2024)
Codificación	Asistentes de código inteligentes para recomendaciones contextuales y generación automática de código.	Haghsheno, S. (2020)
Pruebas	Generación automatizada de casos de prueba, predicción de defectos y monitoreo de pruebas mediante algoritmos de aprendizaje automático.	Haghsheno, S. (2020)
Mantenimiento	Técnicas de reparación automática de programas y refactorización basada en análisis semántico.	Haghsheno, S. (2020)
Despliegue	Sistemas auto-adaptativos impulsados por aprendizaje por refuerzo para optimizar recursos y garantizar estabilidad.	Haghsheno, S. (2020)
Codificación	Generadores automáticos de código basados en aprendizaje profundo y optimización de código.	Ajiga et al., 2024
Pruebas	Algoritmos de aprendizaje automático para predicción de errores y herramientas de pruebas automatizadas.	Ajiga et al., 2024



Mantenimiento	Sistemas predictivos para análisis de impacto y resolución proactiva de problemas.	Ajiga et al., 2024
Despliegue	Herramientas de CI/CD impulsadas por AI para integración y despliegue continuo.	Ajiga et al., 2024
Diseño	Herramientas de diseño automatizado para arquitecturas de software, como DesignBot.	Zhang, 2024
Desarrollo	Generadores automáticos de código como OpenAI Codex y GitHub Copilot; herramientas de detección de errores como DeepCode.	Zhang, 2024
Pruebas	Herramientas de generación de casos de prueba automatizados como Test.AI; herramientas de análisis predictivo de errores.	Zhang, 2024
Mantenimiento	Sistemas predictivos para mantenimiento preventivo y herramientas de resolución automatizada de problemas.	Zhang, 2024
Planificación	Plataformas de gestión impulsadas por AI como Monday.com y Wrike para estimación de esfuerzos y seguimiento de proyectos.	Zhang, 2024
Planificación	Algoritmos de procesamiento de lenguaje natural (NLP) para analizar y extraer requisitos de sistemas complejos.	Siddique, 2022
Pruebas	Sistemas automatizados de validación impulsados por aprendizaje automático para verificar consistencia y precisión.	Siddique, 2022
Planificación	Herramientas de análisis predictivo para estimar recursos y tiempos necesarios en proyectos de ingeniería de sistemas.	Siddique, 2022
Mantenimiento	Modelos de AI explicable para identificar vulnerabilidades y mejorar la seguridad del código generado.	Siddique, 2022
Codificación	Generadores automáticos de código como OpenAI Codex y GPT-4.	Madupati, 2025
Pruebas	Marcos de pruebas inteligentes impulsados por AI para detección de errores y vulnerabilidades.	Madupati, 2025
Despliegue	Pipelines de integración y despliegue continuo (CI/CD) automatizados con IA.	Madupati, 2025
Mantenimiento	Sistemas predictivos basados en AI para monitoreo proactivo y resolución de problemas.	Madupati, 2025
Diseño	Herramientas de optimización de arquitecturas como ArchiMate AI.	Alenezi & Akour, 2025



Desarrollo	Generadores automáticos de código como GitHub Copilot y GPT-4.	Alenezi & Akour, 2025
Pruebas	Plataformas de pruebas automatizadas impulsadas por IA, como Test.AI.	Alenezi & Akour, 2025
Mantenimiento	Sistemas predictivos para monitoreo proactivo y resolución de problemas, como DeepMind Predictive Systems.	Alenezi & Akour, 2025
Planificación	Herramientas de análisis predictivo para estimación de recursos y tiempos, como Monday.com AI.	Alenezi & Akour, 2025
Codificación	Herramientas de revisión automática de código como DeepCode.	Vadde & Munagandla, 2022
Pruebas	Frameworks de pruebas automatizadas impulsados por IA, como Test.AI.	Vadde & Munagandla, 2022
Despliegue	Pipelines CI/CD automatizados con IA, como Jenkins AI.	Vadde & Munagandla, 2022
Mantenimiento	Sistemas predictivos para monitoreo proactivo y resolución de problemas, como Dynatrace AI.	Vadde & Munagandla, 2022
Requisitos	NLP para análisis de requisitos, como IBM Watson y Microsoft Azure NLP.	Upadhyaya, 2022
Diseño	Sketch2Code para convertir bocetos en código HTML.	Upadhyaya, 2022
Codificación	Generadores automáticos de código como GitHub Copilot y Tabnine.	Upadhyaya, 2022
Pruebas	Testim y Appltools para automatización y mantenimiento de casos de prueba.	Upadhyaya, 2022
Despliegue	Herramientas de monitoreo predictivo como Dynatrace y New Relic.	Upadhyaya, 2022
Requisitos	Herramientas de procesamiento de lenguaje natural (NLP) como IBM Watson y Microsoft Azure NLP.	Kokol, 2024
Diseño	Sketch2Code para la generación automática de diseños y Adobe XD con capacidades de optimización de diseño.	Kokol, 2024
Codificación	Generadores automáticos de código como GitHub Copilot y Facebook TransCoder para traducción entre lenguajes.	Kokol, 2024
Pruebas	Testim y Appltools para automatización de pruebas y detección de errores.	Kokol, 2024
Despliegue	Herramientas de monitoreo predictivo como Dynatrace y New Relic para análisis de rendimiento en tiempo real.	Kokol, 2024



Requisitos	Herramientas de procesamiento de lenguaje natural (NLP) como IBM Watson y Microsoft Azure NLP.	Sergeyuk et al., 2025
Diseño	Sketch2Code para la generación automática de diseños y Adobe XD con capacidades de optimización de diseño.	Sergeyuk et al., 2025
Codificación	Generadores automáticos de código como GitHub Copilot y Facebook TransCoder para traducción entre lenguajes.	Sergeyuk et al., 2025
Pruebas	Testim y Applitools para automatización de pruebas y detección de errores.	Sergeyuk et al., 2025
Despliegue	Herramientas de monitoreo predictivo como Dynatrace y New Relic para análisis de rendimiento en tiempo real.	Sergeyuk et al., 2025
Requisitos	ChatGPT para interpretación de requisitos textuales y generación de ideas.	Özpolat et al., 2023
Codificación	ChatGPT para generación automática de código y asistencia en programación.	Özpolat et al., 2023
Pruebas	Herramientas de prueba automatizadas impulsadas por IA, como ChatGPT para generación de casos de prueba.	Özpolat et al., 2023
Mantenimiento	ChatGPT para monitoreo predictivo y análisis de rendimiento en tiempo real.	Özpolat et al., 2023
Requisitos	Herramientas de procesamiento de lenguaje natural (NLP) para interpretar requisitos textuales complejos.	Medepalli, 2025
Diseño	Herramientas generativas para la creación automatizada de arquitecturas de software.	Medepalli, 2025
Codificación	Generadores automáticos de código y asistentes de programación impulsados por IA.	Medepalli, 2025
Pruebas	Frameworks de pruebas automatizadas basados en AI para detectar errores y optimizar procesos de validación.	Medepalli, 2025
Mantenimiento	Modelos predictivos para monitoreo en tiempo real y prevención de fallos en sistemas heredados.	Medepalli, 2025
Pruebas	Redes neuronales de retropropagación (BP Neural Networks), Clasificador Naïve-Bayes, Identificación de huellas digitales, Árboles de decisión	Todorov (2022)
Codificación	Redes neuronales, Algoritmos genéticos, Sistemas de seguridad basados en lógica difusa	Todorov (2022)



Pruebas	Generación inteligente de casos de prueba, Sistemas de prueba auto-reparables, Pruebas visuales	Todorov (2022)
Mantenimiento	Refactorización automática de código, Modelos predictivos para detección de defectos, Sistemas de mantenimiento adaptativos	Todorov (2022)
Planificación	ChatGPT, Bard, Natural Language Processing (NLP), Machine Learning (ML)	Pothukuchi et al. (2023)
Diseño	Generación de patrones de arquitectura, análisis de debilidades en diseños, herramientas de mockups UX	Pothukuchi et al. (2023)
Desarrollo	Generación de código con optimización algorítmica, depuración automática	Pothukuchi et al. (2023)
Pruebas	Generación automática de casos de prueba, validación de datos, identificación de casos límite	Pothukuchi et al. (2023)
Despliegue	Automatización de instalación, monitoreo post-despliegue	Pothukuchi et al. (2023)
Mantenimiento	Diagnóstico proactivo, generación de reportes personalizados, monitoreo continuo	Pothukuchi et al. (2023)
Planificación	ChatGPT (para generación de requisitos y documentación inicial)	Ricárdez et al. (2024)
Diseño	GitHub Copilot (para generación de estructuras de código)	Ricárdez et al. (2024)
Despliegue	GitHub Copilot, Tabnine (para autocompletar código y sugerencias en tiempo real)	Ricárdez et al. (2024)
Pruebas	ChatGPT (para generación de casos de prueba), AI de depuración automática	Ricárdez et al. (2024)
Mantenimiento	ChatGPT (para refactorización y documentación automática de código)	Ricárdez et al. (2024)
Planificación	GitHub Copilot, Modelos Generativos basados en Codex	Upadhyaya (2023)
Diseño	Herramientas de generación de patrones de diseño, Análisis de debilidades en diseño	Upadhyaya (2023)
Desarrollo	GitHub Copilot, Automatización de generación de código seguro	Upadhyaya (2023)
Pruebas	Validación automática de seguridad, Generación de casos de prueba con IA	Upadhyaya (2023)
Despliegue	Automatización de configuraciones, Monitoreo post-despliegue	Upadhyaya (2023)



Mantenimiento	Diagnóstico proactivo con IA, Generación de reportes personalizados	Upadhyaya (2023)
Planificación	Generative AI para análisis de datos, modelos de lenguaje como GPT para generación de ideas.	Russo (2024)
Diseño	Herramientas de generación de patrones de diseño, modelos de lenguaje para propuestas arquitectónicas.	Russo (2024)
Desarrollo	GitHub Copilot, AlphaCode, Amazon CodeWhisperer, Tabnine para generación y refactorización de código.	Russo (2024)
Pruebas	Generación automática de casos de prueba, validación de código con modelos de lenguaje.	Russo (2024)
Despliegue	Automatización de despliegues, monitoreo de anomalías con AI generativa.	Russo (2024)
Mantenimiento	Diagnóstico predictivo, generación de reportes personalizados con modelos de lenguaje.	Russo (2024)
Planificación	JIRA Automation para la gestión de proyectos y priorización de tareas.	Shah & Trehan (2024)
Diseño	Herramientas como GitHub Copilot para generación de prototipos y diseño arquitectónico.	Shah & Trehan (2024)
Desarrollo	Tabnine y GitHub Copilot para generación y refactorización de código.	Shah & Trehan (2024)
Pruebas	Testim.io y AppliTools para pruebas automatizadas y mejora de cobertura de pruebas.	Shah & Trehan (2024)
Despliegue	Herramientas como Automation Anywhere para optimización de despliegues.	Shah & Trehan (2024)
Mantenimiento	SonarQube para análisis estático de código y detección de errores.	Shah & Trehan (2024)
Planificación	Herramientas de análisis predictivo para identificar tendencias y necesidades organizacionales.	Al-kfairy (2025)
Diseño	Generative AI para la creación de prototipos y simulaciones rápidas.	Al-kfairy (2025)
Desarrollo	Plataformas como GitHub Copilot para generación y optimización de código.	Al-kfairy (2025)
Pruebas	Herramientas de pruebas automatizadas basadas en AI para detectar errores y vulnerabilidades.	Al-kfairy (2025)



Despliegue	Sistemas de monitoreo impulsados por AI para garantizar estabilidad post-despliegue.	Al-kfairy (2025)
Mantenimiento	Soluciones de diagnóstico predictivo para anticipar fallos y optimizar el rendimiento.	Al-kfairy (2025)
Planificación	Herramientas de NLP para traducir lenguaje natural en especificaciones técnicas.	Keskar (2023)
Diseño	Generative AI para crear diagramas UML, wireframes y mockups.	Keskar (2023)
Desarrollo	GitHub Copilot y Tabnine para generación y optimización de código.	Keskar (2023)
Pruebas	Herramientas de pruebas automatizadas basadas en AI para generar matrices de prueba.	Keskar (2023)
Despliegue	Sistemas de monitoreo impulsados por AI para supervisar entornos de despliegue.	Keskar (2023)
Mantenimiento	Algoritmos predictivos para generación automática de parches y actualizaciones.	Keskar (2023)
Planificación	Herramientas de NLP para analizar documentos de requisitos y estructurarlos automáticamente.	Shao et al. (2025)
Diseño	IA para generar propuestas de diseño arquitectónico y evaluar rendimiento.	Shao et al. (2025)
Desarrollo	Agentes de desarrollo para generación y revisión de código.	Shao et al. (2025)
Pruebas	Herramientas de generación de casos de prueba inteligentes basadas en IA.	Shao et al. (2025)
Despliegue	Sistemas de monitoreo impulsados por AI para supervisar despliegues en tiempo real.	Shao et al. (2025)
Mantenimiento	Agentes de gestión del conocimiento con RAG para actualizaciones dinámicas.	Shao et al. (2025)
Planificación	Herramientas de NLP para transformar lenguaje natural en especificaciones técnicas.	Oluwaseun (2023)
Diseño	Simuladores basados en AI para evaluar arquitecturas y diseños adaptativos.	Oluwaseun (2023)
Desarrollo	Generadores de código como GitHub Copilot para sugerencias y creación de funciones.	Oluwaseun (2023)
Pruebas	Herramientas de pruebas automatizadas impulsadas por machine learning.	Oluwaseun (2023)
Despliegue	Sistemas de monitoreo predictivo basados en AI para supervisión en tiempo real.	Oluwaseun (2023)



UNIVERSIDAD
NACIONAL
DE LA PLATA



**“Desde los Obstáculos hasta las Estrategias: Integración
de la inteligencia artificial en el ciclo de vida del
desarrollo del software.”**

Mantenimiento	Algoritmos de mantenimiento predictivo para anticipar fallos y optimizar recursos.	Oluwaseun (2023)
---------------	--	------------------



Anexo 2: Sistematización de la Información Extraída

Este anexo presenta la sistematización de los datos recopilados durante la fase de revisión bibliográfica. La información se organizó en una hoja de cálculo con el propósito de clasificar y sintetizar los hallazgos de la literatura respecto a la integración de la AI en el SDLC. De esta manera, la matriz proporciona una visión estructurada y detallada de los desafíos, factores, estrategias e impactos reportados en los estudios seleccionados.

Estructura de la Hoja de Cálculo

La hoja de cálculo se compone de las siguientes columnas, cada una de las cuales recoge un aspecto particular de la información relevante extraída de los artículos analizados:

1. Fase del Ciclo de Vida

En esta columna se indica la etapa específica del SDLC a la que se refieren los hallazgos del estudio.

- **Propósito:** Permite agrupar la información según el momento del proceso de desarrollo (por ejemplo, planificación, diseño, desarrollo, pruebas, despliegue y mantenimiento) y facilita el análisis comparativo entre las diferentes fases.

2. Desafío Identificado

Aquí se registran los obstáculos o problemas que los estudios han señalado en cada fase del SDLC.

- **Propósito:** Resalta las dificultades concretas que afectan la integración de la IA, como la falta de datos precisos, problemas al automatizar ciertos procesos o limitaciones en la generación de código. Este registro es fundamental para identificar patrones y tendencias en los desafíos presentados.

3. Factor Contribuyente

Esta columna explica las causas o circunstancias que generan o agravan los desafíos identificados.

- **Propósito:** Permite entender el origen de los problemas (por ejemplo, ausencia de modelos predictivos, resistencia a la adopción tecnológica o limitaciones en las



herramientas tradicionales). Con esta información se pueden establecer conexiones entre los desafíos y sus raíces, lo cual es clave para el diseño de estrategias de solución.

4. Estrategia Sugerida

En esta sección se anotan las soluciones o recomendaciones que proponen los autores para superar los desafíos detectados en cada fase del SDLC.

- **Propósito:** Ayuda a compilar las propuestas de mitigación y optimización, como el uso de algoritmos de aprendizaje automático, la implementación de plataformas interoperables o la capacitación continua. Esta columna sirve de base para el análisis comparativo de las estrategias y para identificar aquellas más efectivas o innovadoras.

5. Impacto en la Eficiencia o Calidad

Aquí se resume el efecto esperado o reportado por cada estrategia en términos de mejora en la eficiencia o calidad del proceso de desarrollo.

- **Propósito:** Permite evaluar la relevancia práctica de las estrategias sugeridas, ya que se cuantifica o describe cómo la solución contribuye a mejorar aspectos críticos del SDLC (por ejemplo, reducción de tiempo de desarrollo, mayor cobertura de pruebas o mejora en la precisión de las estimaciones).

6. Referencia

Esta columna indica la fuente bibliográfica de donde se obtuvo la información registrada en las otras columnas.

- **Propósito:** Garantiza la trazabilidad de la información, permitiendo al lector consultar directamente los estudios originales y verificar la veracidad y relevancia de cada hallazgo. Además, facilita la integración de las citas en el marco teórico y las demás secciones del trabajo académico.

Aplicaciones Prácticas



- **Identificación de Patrones:** La matriz facilita la detección de tendencias comunes y vacíos en la literatura, evidenciando qué fases del SDLC presentan mayores desafíos y cuáles son las estrategias más recurrentes y prometedoras para superarlos.
- **Soporte para el Marco Teórico:** La sistematización de la información respalda la argumentación en el marco teórico, ya que permite relacionar los hallazgos con las propuestas y debates existentes en la literatura.
- **Base para Propuestas de Estrategia:** Al integrar los desafíos, factores contributivos y estrategias sugeridas, la matriz se convierte en una herramienta esencial para formular recomendaciones prácticas y fundamentadas que se desarrollarán en la sección de recomendaciones.

Tabla 2. Matriz de Extracción de Datos: Desafíos, Factores Contribuyentes, Estrategias Sugeridas e Impacto en el SDLC

Fase del Ciclo de Vida	Desafío Identificado	Factor Contribuyente	Estrategia Sugerida	Impacto en la Eficiencia o Calidad	Referencia
Planificación	Complejidad en la estimación del esfuerzo del proyecto	Falta de datos precisos y modelos predictivos robustos	Uso de modelos de aprendizaje automático (ML)	Mejora en la precisión de las estimaciones y optimización de recursos	Durrani et al., 2024
Diseño	Dificultad para automatizar el diseño arquitectónico	Limitaciones en las herramientas y enfoques tradicionales	Implementación de algoritmos de optimización	Reducción de tiempo en diseño arquitectónico y mejora en la precisión	Durrani et al., 2024
Desarrollo	Identificación temprana de problemas de seguridad	Dependencia de detección manual y carencia de herramientas eficaces	Uso de ML para detección proactiva	Detección temprana de vulnerabilidades y reducción de errores en el código	Durrani et al., 2024
Pruebas	Cobertura insuficiente en casos de	Recursos limitados y métodos	Generación automatizada de casos de	Cobertura de pruebas más amplia y	Durrani et al., 2024



	prueba	tradicionales	prueba con IA	reducción del tiempo requerido	
Despliegue y Mantenimiento	Predicción y prevención limitada de defectos	Modelos complejos de mantenimiento y falta de automatización	IA explicable y herramientas de predicción avanzada	Mayor confiabilidad en las operaciones de mantenimiento y mejor calidad del software	Durrani et al., 2024
Planificación	Complejidad en la integración de herramientas y tecnologías	Falta de compatibilidad y estándares uniformes	Uso de herramientas interoperables y plataformas estandarizadas	Mejora en la coordinación entre equipos	Kambala, G. (2024)
Desarrollo	Priorización y automatización de pruebas	Falta de datos relevantes o inconsistencias en el código	Aplicación de algoritmos de aprendizaje automático para priorización	Reducción de tiempos y mayor cobertura en pruebas	Kambala, G. (2024)
Despliegue	Gestión de escalabilidad dinámica	Recursos insuficientes o mal distribuidos	Uso de aprendizaje reforzado y escalado automático de infraestructura	Garantía de alto rendimiento y disponibilidad	Kambala, G. (2024)
Mantenimiento	Detección de anomalías y vulnerabilidades	Datos ruidosos y modelos inadecuados	Implementación de sistemas de monitoreo basado en analítica predictiva	Resolución proactiva de problemas antes de impactar al usuario final	Kambala, G. (2024)
Planificación	Familiaridad y entrenamiento en el uso de	Falta de capacitación y experiencia	Programas de capacitación extensiva y	Mejora en la adaptación y aprovechamiento	Giffari et al. (2024)



	herramientas	previa	transferencia de conocimiento	nto de herramientas	
Planificación	Calidad y credibilidad de los datos generados	Falta de precisión y validación de resultados	Supervisión humana para validar resultados y ajustar modelos	Incremento en la confianza y precisión del análisis	Giffari et al. (2024)
Planificación	Dependencia excesiva de las herramientas	Pérdida de habilidades críticas y adaptabilidad	Uso equilibrado de herramientas como soporte y no como sustituto	Mitigación de riesgos ante cambios en estándares o sistemas	Giffari et al. (2024)
Planificación	Seguridad y privacidad de los datos	Vulnerabilidades en herramientas de IA	Políticas robustas de seguridad y evaluación de herramientas antes de uso	Protección de datos sensibles y confianza de los analistas	Giffari et al. (2024)
Planificación	Falta de necesidad organizacional	Decisiones estratégicas limitadas por la alta gestión	Evaluación de necesidades y urgencia antes de adoptar tecnologías	Optimización de recursos en la adopción tecnológica	Giffari et al. (2024)
Planificación	Estimaciones imprecisas del tamaño del proyecto	Falta de modelos predictivos sólidos	Uso de modelos de aprendizaje profundo (<i>Deep Learning</i>)	Mejora en la precisión de las estimaciones y en la asignación de recursos	Benitez & Serrano (2023)
Planificación	Incertidumbre en los requisitos del software	Falta de herramientas para análisis y predicción	Implementación de <i>Machine Learning</i> para especificaciones de requisitos y predicción de	Incremento en la calidad y precisión de los requisitos	Benitez & Serrano (2023)



			vulnerabilidades		
Diseño	Identificación tardía de errores	Limitaciones en el modelado tradicional	Uso de <i>Natural Language Processing</i> para generación de diagramas y redes neuronales para predicción de errores	Reducción de errores en la etapa de diseño	Benitez & Serrano (2023)
Implementación	Puntos débiles en el código fuente	Revisiones manuales limitadas	Generación de código con AI generativa y detección de errores con <i>Machine Learning</i>	Mejora en la calidad del software y menor tiempo de desarrollo	Benitez & Serrano (2023)
Pruebas	Cobertura insuficiente de casos de prueba	Falta de automatización	Automatización de escenarios y clasificación de casos de prueba con aprendizaje profundo y NLP	Mayor cobertura y precisión en la validación	Benitez & Serrano (2023)
Mantenimiento	Ineficiencia en la predicción de mantenimiento	Modelos predictivos limitados	Uso de <i>Machine Learning</i> para detección de deuda técnica y aprendizaje profundo para predicción de métricas	Mejora en la sostenibilidad del software	Benitez & Serrano (2023)



Planificación	Dificultad para estimar recursos y tiempos	Modelos de AI poco precisos	Mejorar los modelos predictivos mediante aprendizaje profundo	Incremento en la precisión de las estimaciones y optimización de recursos	Khan, Fatima & Ahmed (2023)
Planificación	Calidad inconsistente de los datos	Falta de datasets de calidad	Implementar procesos de validación y limpieza de datos	Mejora en la confiabilidad de los resultados	Khan, Fatima & Ahmed (2023)
Diseño	Resistencia organizacional	Falta de apoyo gerencial y resistencia al cambio	Promover la capacitación y el compromiso de los stakeholders	Incremento en la aceptación y adopción de herramientas IA	Khan, Fatima & Ahmed (2023)
Implementación	Adaptabilidad limitada de los equipos	Falta de habilidades técnicas	Programas de formación continua para desarrolladores	Mejora en la implementación y calidad del código	Khan, Fatima & Ahmed (2023)
Pruebas	Limitaciones técnicas en la automatización	Complejidad de los algoritmos	Simplificar los algoritmos y proporcionar herramientas accesibles	Mayor cobertura y eficiencia en las pruebas	Khan, Fatima & Ahmed (2023)
Mantenimiento	Preocupaciones éticas sobre privacidad de datos	Riesgos de seguridad en el manejo de datos	Establecer políticas de seguridad y estándares éticos	Incremento en la confianza y sostenibilidad del software	Khan, Fatima & Ahmed (2023)
Planificación	Sesgos en los datos utilizados para entrenar modelos	Intervención humana en la curación de datos	Implementar procesos de validación y limpieza de datos	Mejora en la precisión y equidad de los resultados	Garg, K. (2023)
Diseño	Selección de algoritmos adecuados	Complejidad en la variedad de algoritmos	Capacitación para seleccionar algoritmos óptimos	Aceleración del proceso de diseño y desarrollo	Garg, K. (2023)



Implementación	Vulnerabilidad a ataques adversariales	Manipulación de entradas para engañar modelos	Monitoreo continuo y robustos mecanismos de seguridad	Incremento en la confiabilidad y seguridad del software	Garg, K. (2023)
Pruebas	Escalabilidad de los algoritmos	Aumento en el volumen de datos y usuarios	Diseñar algoritmos eficientes y adaptables	Mejora en el rendimiento bajo alta demanda	Garg, K. (2023)
Mantenimiento	Impacto ambiental de los modelos de IA	Alto consumo energético en el entrenamiento	Desarrollo de modelos más eficientes energéticamente	Reducción del impacto ambiental y sostenibilidad	Garg, K. (2023)
Planificación	Compatibilidad con sistemas heredados	Diferencias arquitectónicas entre sistemas	Diseñar interfaces estándar y adoptar arquitecturas modulares	Mejora en la integración y reducción de costos	Kumar, S. (2024)
Planificación	Complejidad en la extracción de requisitos	Datos no estructurados y ambiguos	Uso de técnicas de NLP para analizar retroalimentación y documentación	Incremento en la precisión y cobertura de requisitos	Kumar, S. (2024)
Codificación	Brecha de habilidades	Escasez de profesionales capacitados en IA	Implementar programas de capacitación y colaboración entre expertos	Mejora en la calidad del código y reducción de errores	Kumar, S. (2024)
Pruebas	Detección de anomalías	Falta de cobertura en escenarios de prueba	Uso de aprendizaje no supervisado para identificar comportamientos	Incremento en la confiabilidad del software	Kumar, S. (2024)



			inesperados		
Despliegue	Complejidad en el despliegue automatizado	Dependencias entre componentes del sistema	Implementar procesos de auto-recuperación y monitoreo continuo	Reducción de fallos y mejora en la estabilidad del sistema	Kumar, S. (2024)
Codificación	Explicabilidad de los modelos de IA	Naturaleza de "caja negra" de los modelos	Uso de técnicas de AI explicable como mapas de atención y sistemas basados en reglas	Mejora en la confianza y adopción de herramientas de IA	Müller, M. (2024)
Pruebas	Calidad y disponibilidad de datos	Datos fragmentados e inconsistentes	Implementar prácticas robustas de gestión y curación de datos	Incremento en la cobertura y precisión de las pruebas	Müller, M. (2024)
Mantenimiento	Integración con sistemas existentes	Compatibilidad técnica y resistencia organizacional	Desarrollo de herramientas compatibles y capacitación del personal	Reducción de fallos y mejora en la sostenibilidad del software	Müller, M. (2024)
Despliegue	Escalabilidad de los sistemas	Complejidad en la adaptación a entornos dinámicos	Uso de sistemas auto-adaptativos y aprendizaje por refuerzo	Mejora en la estabilidad y rendimiento del sistema	Müller, M. (2024)
Codificación	Explicabilidad de los modelos de IA	Naturaleza de "caja negra" de los modelos	Desarrollo de técnicas de AI explicable como mapas de atención y modelos basados en reglas	Mejora en la confianza y adopción de herramientas de IA	Haghsheno, S. (2020)



Pruebas	Calidad y disponibilidad de datos	Datos fragmentados e inconsistentes	Implementar prácticas robustas de gestión y curación de datos	Incremento en la cobertura y precisión de las pruebas	Haghsheno, S. (2020)
Mantenimiento	Integración con sistemas existentes	Compatibilidad técnica y resistencia organizacional	Adopción de arquitecturas modulares y capacitación del personal	Reducción de fallos y mejora en la sostenibilidad del software	Haghsheno, S. (2020)
Despliegue	Escalabilidad de los sistemas	Complejidad en la adaptación a entornos dinámicos	Uso de sistemas auto-adaptativos y aprendizaje por refuerzo	Mejora en la estabilidad y rendimiento del sistema	Haghsheno, S. (2020)
Codificación	Complejidad de los modelos de IA	Requieren experiencia técnica avanzada	Capacitación en AI para desarrolladores	Mejora la eficiencia y reduce errores	Ajiga et al., 2024
Pruebas	Privacidad y seguridad de los datos	Manejo de grandes volúmenes de información	Implementar marcos robustos de gobernanza de datos	Incrementa la confianza y seguridad de las pruebas	Ajiga et al., 2024
Mantenimiento	Integración con sistemas heredados	Compatibilidad técnica y resistencia organizacional	Desarrollar soluciones modulares y flexibles	Facilita la actualización y reduce riesgos técnicos	Ajiga et al., 2024
Despliegue	Resistencia organizacional	Falta de capacitación continua en herramientas de IA	Fomentar una cultura de aprendizaje continuo	Mejora la adopción de AI y reduce tiempos de respuesta	Ajiga et al., 2024
Diseño	Reconocimiento limitado de patrones de diseño	Falta de herramientas avanzadas de análisis	Implementación de AI que analice arquitecturas existentes	Mejora la estructuración y escalabilidad del sistema	Zhang, 2024



Desarrollo	Dependencia excesiva en herramientas de generación automática	Pérdida de experiencia técnica entre los desarrolladores	Capacitación continua y combinación de AI con procesos manuales	Reduce la dependencia en herramientas automáticas	Zhang, 2024
Pruebas	Vulnerabilidades en el código generado por IA	Algoritmos que priorizan rapidez sobre seguridad	Uso de técnicas de AI explicable para pruebas de seguridad	Incrementa la confiabilidad y transparencia del software	Zhang, 2024
Mantenimiento	Dificultades en predicción de fallas críticas	Falta de modelos predictivos específicos	Implementación de sistemas predictivos y auto-corrección	Reduce el tiempo de inactividad y mejora la sostenibilidad	Zhang, 2024
Planificación	Subestimación de recursos y tiempos necesarios	Análisis basado en datos históricos limitados	Mejora de herramientas predictivas para estimación	Incrementa la eficiencia de la gestión de proyectos	Zhang, 2024
Planificación	Complejidad en la extracción de requisitos	Ambigüedades e inconsistencias en los datos	Uso de algoritmos de aprendizaje automático y procesamiento de lenguaje natural	Mejora la precisión y consistencia en los requisitos	Siddique, 2022
Pruebas	Limitaciones en la verificación de resultados	Falta de procesos robustos de validación	Implementación de sistemas automatizados de validación	Incrementa la confiabilidad y precisión del software	Siddique, 2022
Planificación	Subestimación de recursos y tiempos necesarios	Análisis limitado de datos históricos	Herramientas impulsadas por AI para estimación y planificación	Mejora la eficiencia en la gestión de proyectos	Siddique, 2022
Despliegue y mantenimiento	Vulnerabilidades en el	Algoritmos que priorizan	Técnicas de AI explicable	Incrementa la seguridad y	Siddique, 2022



to	código generado por IA	rapidez sobre seguridad	para análisis de seguridad	confiabilidad del sistema	
Planificación	Ambigüedad en la interpretación de requisitos iniciales	Datos insuficientes o inconsistentes	Implementaci ón de algoritmos de NLP para aclarar y estructurar los requisitos	Mejora la comprensión y reduce errores tempranos	Madupati, 2025
Codificación	Dependencia excesiva en herramientas de generación automática de código	Pérdida de habilidades manuales entre los desarrolladore s	Capacitación continua para equilibrar habilidades manuales e IA	Incrementa la calidad y reduce riesgos técnicos	Madupati, 2025
Pruebas	Identificación incompleta de vulnerabilidad es críticas	Enfoque limitado en seguridad en pruebas automatizadas	Desarrollo de frameworks avanzados de pruebas impulsados por IA	Incrementa la seguridad y confiabilidad del software	Madupati, 2025
Despliegue	Dificultades en la integración con metodologías tradicionales como Agile y Waterfall	Incompatibilid ad entre procesos manuales y automatizados	Uso de pipelines CI/CD automatizado s para facilitar la integración	Reduce tiempos y mejora la eficiencia del despliegue	Madupati, 2025
Mantenimie nto	Falta de modelos predictivos avanzados para la identificación de fallos	Datos históricos insuficientes para predicciones precisas	Implementaci ón de sistemas predictivos de AI para monitoreo proactivo	Incrementa la sostenibilidad y reduce el tiempo de inactividad	Madupati, 2025
Diseño	Falta de estandarizació n en herramientas de IA	Ausencia de estándares universales	Desarrollo de marcos estandarizado s	Mejora la interoperabili dad y reduce errores	Alenezi & Akour, 2025



Desarrollo	Dependencia excesiva en herramientas de generación automática	Pérdida de habilidades técnicas entre desarrolladores	Capacitación continua para equilibrar habilidades manuales e IA	Incrementa la calidad del código y reduce errores	Alenezi & Akour, 2025
Pruebas	Complejidad en la generación de casos de prueba automatizados	Algoritmos que priorizan rapidez sobre exhaustividad	Uso de técnicas avanzadas de AI para pruebas de seguridad	Incrementa la confiabilidad y seguridad del software	Alenezi & Akour, 2025
Mantenimiento	Dificultades en la integración con sistemas heredados	Falta de compatibilidad técnica	Implementación de soluciones modulares y flexibles	Facilita la actualización y reduce riesgos técnicos	Alenezi & Akour, 2025
Planificación	Resistencia organizacional	Barreras culturales y técnicas	Capacitación y adopción gradual de herramientas de IA	Mejora la adopción y reduce tiempos de implementación	Alenezi & Akour, 2025
Codificación	Complejidad en la revisión automática de código	Falta de algoritmos avanzados para análisis de código	Implementación de herramientas de AI para revisión automática de código	Mejora la calidad del código y reduce errores humanos	Vadde & Munagandla, 2022
Pruebas	Detección limitada de vulnerabilidades críticas	Enfoque insuficiente en seguridad en pruebas automatizadas	Uso de frameworks de pruebas impulsados por AI para priorizar casos críticos	Incrementa la seguridad y confiabilidad del software	Vadde & Munagandla, 2022
Despliegue	Dificultades en la integración con sistemas heredados	Falta de compatibilidad técnica	Automatización de pipelines CI/CD con AI para facilitar la integración	Reduce tiempos de despliegue y mejora la eficiencia	Vadde & Munagandla, 2022



Mantenimiento	Respuesta lenta a incidentes críticos	Falta de sistemas predictivos para monitoreo	Implementación de AI para detección proactiva de fallos y resolución automática	Incrementa la sostenibilidad y reduce tiempos de inactividad	Vadde & Munagandla, 2022
Planificación	Ambigüedad en la interpretación de requisitos	Falta de estructura en los datos textuales	Uso de herramientas NLP como IBM Watson para estructurar y priorizar requisitos	Mejora la comprensión y evita errores tempranos	Upadhyaya, 2022
Diseño	Dificultad para traducir ideas en diseños funcionales	Limitaciones en la automatización de diseño	Implementación de herramientas como Sketch2Code para convertir bocetos en código	Acelera la creación de diseños y optimiza recursos	Upadhyaya, 2022
Codificación	Complejidad en la generación automática de código	Dependencia de grandes volúmenes de datos para entrenamiento	Uso de asistentes de código como GitHub Copilot o Tabnine para sugerencias de código	Reduce tiempos de desarrollo y mejora la productividad	Upadhyaya, 2022
Pruebas	Limitaciones en la identificación de errores críticos	Falta de adaptabilidad en casos de prueba	Uso de herramientas como Testim para crear y mantener casos de prueba adaptativos	Incrementa la calidad del software y la confiabilidad	Upadhyaya, 2022
Despliegue	Respuesta limitada a anomalías en	Dependencia de monitoreo manual	Implementación de herramientas	Mejora la estabilidad del sistema y	Upadhyaya, 2022



	tiempo real		como Dynatrace para monitoreo predictivo	reduce tiempos de inactividad	
Planificación	Ambigüedad en la interpretación de requisitos	Falta de estandarización en herramientas de IA	Uso de NLP para estructurar y priorizar requisitos	Mejora la claridad y evita errores tempranos	Kokol, 2024
Diseño	Dificultad para optimizar arquitecturas	Limitaciones en la automatización de diseño	Implementación de herramientas de AI para sugerir patrones óptimos	Acelera la creación de diseños y mejora la calidad	Kokol, 2024
Codificación	Complejidad en la generación automática de código	Dependencia de datos de alta calidad	Uso de asistentes de código como GitHub Copilot	Incrementa la productividad y reduce errores humanos	Kokol, 2024
Pruebas	Limitaciones en la detección de errores críticos	Falta de adaptabilidad en casos de prueba	Uso de herramientas de pruebas automatizadas impulsadas por IA	Incrementa la confiabilidad y seguridad del software	Kokol, 2024
Mantenimiento	Dificultades en la integración con sistemas heredados	Falta de compatibilidad técnica	Implementación de sistemas predictivos para monitoreo proactivo	Reduce tiempos de inactividad y mejora la sostenibilidad	Kokol, 2024
Requisitos	Ambigüedad en la interpretación de requisitos	Falta de explicabilidad en herramientas de IA	Desarrollo de modelos explicables para mejorar la comprensión	Mejora la confianza y reduce errores en etapas iniciales	Sergeyuk et al., 2025



Diseño	Dificultad para personalizar arquitecturas	Limitaciones en la adaptabilidad de herramientas	Implementación de herramientas de AI con opciones de personalización	Optimiza arquitecturas y mejora la calidad del diseño	Sergeyuk et al., 2025
Codificación	Dependencia excesiva en herramientas de IA	Sesgo de automatización	Capacitación continua para reducir la dependencia	Incrementa la autonomía de los desarrolladores	Sergeyuk et al., 2025
Pruebas	Sobrecarga de verificación	Falta de mecanismos de validación automatizados	Introducción de sistemas de verificación basados en IA	Incrementa la confiabilidad y reduce la carga de trabajo	Sergeyuk et al., 2025
Mantenimiento	Dificultades en la integración con sistemas heredados	Falta de compatibilidad técnica	Uso de sistemas predictivos para monitoreo proactivo	Reduce tiempos de inactividad y mejora la sostenibilidad	Sergeyuk et al., 2025
Requisitos	Limitaciones en la interpretación de requisitos complejos	Dependencia de la comprensión contextual	Implementación de modelos NLP avanzados como ChatGPT	Mejora la precisión en la recopilación de requisitos	Özpolat et al., 2023
Diseño	Dificultades en la generación automatizada de diseños	Escasez de datos en proyectos específicos	Uso de herramientas con capacidades generativas	Optimiza la creación de arquitecturas funcionales	Özpolat et al., 2023
Codificación	Complejidad en la validación de código generado por IA	Dependencia de datos de alta calidad	Validación humana en conjunto con herramientas automáticas	Incrementa la confiabilidad y reduce errores	Özpolat et al., 2023



Pruebas	Limitaciones en la adaptabilidad de pruebas automatizadas	Falta de mecanismos predictivos robustos	Desarrollo de frameworks específicos impulsados por IA	Mejora la eficiencia en la detección de errores	Özpolat et al., 2023
Mantenimiento	Desafíos en la integración con sistemas heredados	Baja compatibilidad técnica	Implementación de soluciones de monitoreo predictivo	Reduce tiempos de inactividad y asegura la sostenibilidad	Özpolat et al., 2023
Requisitos	Limitaciones en la interpretación de requisitos complejos	Falta de herramientas de AI adaptadas al contexto	Uso de modelos NLP avanzados para mejorar la precisión en la interpretación de requisitos	Mejora la claridad y reduce errores en etapas iniciales	Medepalli, 2025
Diseño	Dificultades en la personalización de arquitecturas	Escasez de datos específicos para entrenar modelos	Implementación de herramientas generativas con capacidades de personalización	Optimiza la calidad del diseño y reduce tiempos	Medepalli, 2025
Codificación	Complejidad en la validación de código generado automáticamente	Dependencia de datos de alta calidad	Validación colaborativa entre humanos y herramientas de IA	Incrementa la confiabilidad y reduce errores	Medepalli, 2025
Pruebas	Limitaciones en la adaptabilidad de pruebas automatizadas	Falta de mecanismos predictivos robustos	Desarrollo de frameworks específicos impulsados por IA	Mejora la eficiencia en la detección de errores	Medepalli, 2025
Mantenimiento	Desafíos en la integración con sistemas	Baja compatibilidad técnica	Uso de modelos predictivos	Reduce tiempos de inactividad y	Medepalli, 2025



	heredados		para monitoreo proactivo	asegura la sostenibilidad	
Planificación	Disponibilidad y calidad de datos insuficientes para entrenar modelos de IA.	Falta de datos históricos en ingeniería de software.	Mejorar la recopilación de datos históricos y procesos de preprocesami ento.	Incrementa la precisión y confiabilidad de los modelos de IA.	Todorov (2022), sección 6.1.
Diseño	Falta de sistemas explicables para interpretar decisiones complejas.	Modelos de AI altamente opacos y no interpretables.	Desarrollar sistemas de AI explicables para facilitar depuración y cumplimiento.	Mejora la confianza en los modelos y facilita la adopción en equipos técnicos.	Todorov (2022), sección 6.1.
Diseño	Problemas de compatibilida d entre herramientas de AI y sistemas existentes.	Infraestructur a de software que no está diseñada para IA.	Modificar procesos para incluir compatibilida d con herramientas basadas en IA.	Aumenta la sostenibilidad y reduce los costos de integración.	Todorov (2022), sección 6.1.
Mantenimie nto	Necesidad de sistemas proactivos que detecten defectos antes de que ocurran.	Ausencia de modelos predictivos y enfoque reactivo.	Implementar mantenimient o predictivo utilizando IA.	Reduce los tiempos de inactividad y mejora la confiabilidad del software.	Todorov (2022), sección 6.2.
Planificación	Ambigüedad en la definición de requisitos y estimaciones inexactas.	Falta de datos históricos y herramientas predictivas.	Uso de AI generativa para analizar datos históricos y generar estimaciones precisas.	Mejora la precisión en la planificación y reduce el tiempo en esta etapa.	Pothukuch i et al. (2023)



Diseño	Dificultades en la selección de patrones de arquitectura y uniformidad.	Complejidad en las decisiones de diseño y falta de documentación adecuada.	Implementación de AI generativa para proponer diseños iniciales y evaluar alternativas.	Incrementa la escalabilidad y la consistencia de los diseños.	Pothukuchi et al. (2023)
Desarrollo	Generación de código poco optimizada y propensa a errores.	Falta de automatización para mejora algorítmica.	IA generativa para sugerir código, optimizar recursos y depurar errores.	Acelera la codificación y mejora la calidad del código producido.	Pothukuchi et al. (2023)
Pruebas	Cobertura limitada y tiempo extenso en la creación manual de pruebas.	Dependencia de scripts de prueba manuales.	Automatización de casos de prueba con AI generativa y análisis automático.	Mejora la cobertura y reduce el tiempo necesario para validar el software.	Pothukuchi et al. (2023)
Despliegue	Complejidad en la organización de tareas de despliegue.	Falta de automatización en la liberación de software.	IA generativa para automatizar tareas de configuración, instalación y monitoreo.	Aumenta la eficiencia del despliegue y reduce errores en producción.	Pothukuchi et al. (2023)
Mantenimiento	Resolución lenta de problemas y diagnóstico reactivo.	Falta de herramientas predictivas y monitoreo continuo.	Uso de AI generativa para diagnóstico proactivo y personalización de reportes.	Incrementa la tasa de resolución y asegura el cumplimiento de estándares.	Pothukuchi et al. (2023)
Planificación	Resistencia al cambio en los equipos de desarrollo.	Falta de capacitación y escepticismo sobre el uso de AI en programación.	Capacitación y adopción progresiva, introduciendo AI en proyectos	Facilita la adopción de AI y reduce la curva de aprendizaje, mejorando la	Ricárdez et al. (2024)



			pequeños antes de escalar su uso.	productividad.	
Diseño	Interoperabili dad y estándares.	Incompatibili dad con herramientas de desarrollo tradicionales.	Definir estándares y buenas prácticas para la integración de AI en el ciclo de vida del software.	Mejora la integración fluida con herramientas existentes, optimizando el desarrollo.	Ricárdez et al. (2024)
Desarrollo	Calidad y exactitud del código generado.	La AI puede generar código incorrecto o ineficiente.	Implementar revisiones manuales y pruebas automatizada s para verificar la calidad del código.	Reduce errores y mejora la fiabilidad del software.	Ricárdez et al. (2024)
Pruebas	Seguridad y privacidad.	Riesgo de exposición de datos sensibles al utilizar herramientas de IA.	Implementar controles de acceso y anonimización de datos.	Protege la información confidencial y reduce vulnerabilidad es.	Ricárdez et al. (2024)
Mantenimie nto	Sesgos y limitaciones de la IA.	Dependencia de datos sesgados en el entrenamient o de la IA.	Monitoreo continuo y ajuste de los modelos de AI para reducir sesgos.	Asegura que las soluciones de AI sean justas y eficientes a largo plazo.	Ricárdez et al. (2024)
Planificación	Calidad limitada de datos de entrenamient o para AI generativa.	Dependencia de repositorios públicos con prácticas inseguras.	Curación continua de datasets para reflejar estándares modernos de seguridad.	Mejora la precisión en la planificación y reduce los riesgos iniciales.	Upadhyay a (2023)



Diseño	Falta de comprensión profunda de contextos específicos de aplicaciones.	Limitaciones contextuales de las herramientas de IA.	Supervisión activa de desarrolladores para validar las sugerencias de diseño.	Incrementa la coherencia y la seguridad de los diseños producidos.	Upadhyaya (2023)
Desarrollo	Generación de código con posibles vulnerabilidades.	Entrenamiento de AI en datos desactualizados.	Validación por parte de desarrolladores y mejoras continuas en los modelos de IA.	Reduce la introducción de vulnerabilidades y mejora la calidad del código.	Upadhyaya (2023)
Pruebas	Detección insuficiente de vulnerabilidades en tiempo real.	Falta de integración de herramientas automatizadas con seguridad.	Uso de herramientas generativas para realizar validaciones automáticas de seguridad.	Incrementa la cobertura de pruebas y reduce el tiempo necesario para detectar fallos.	Upadhyaya (2023)
Despliegue	Riesgos en configuraciones inseguras durante el despliegue.	Falta de automatización en procesos de configuración y monitoreo.	Implementación de AI para automatizar configuraciones y análisis de rendimiento.	Mejora la eficiencia del despliegue y reduce errores en producción.	Upadhyaya (2023)
Mantenimiento	Diagnóstico reactivo y resolución tardía de problemas.	Ausencia de monitoreo proactivo y herramientas predictivas.	Uso de AI generativa para identificar problemas de seguridad y generar reportes personalizados.	Reduce los tiempos de resolución y asegura el cumplimiento de estándares.	Upadhyaya (2023)
Planificación	Compatibilidad limitada con flujos de	Falta de integración efectiva de	Diseñar herramientas adaptativas	Mejora la adopción y reduce la	Russo (2024)



	trabajo existentes.	herramientas de AI generativa.	que se alineen con metodologías actuales.	resistencia organizacional .	
Diseño	Curva de aprendizaje elevada para herramientas de IA.	Complejidad técnica y falta de capacitación adecuada.	Implementar programas de entrenamiento o continuo y diseño centrado en el usuario.	Facilita la adopción y mejora la eficiencia en el diseño.	Russo (2024)
Desarrollo	Dependencia excesiva de herramientas de AI generativa.	Riesgo de pérdida de habilidades fundamentales en programación.	Promover el uso responsable y complementar con revisión humana.	Mantiene la calidad del código y reduce errores.	Russo (2024)
Pruebas	Dificultades en la validación de código generado por IA.	Preocupaciones sobre la precisión y calidad del código.	Establecer mejores prácticas para revisión y validación de código.	Incrementa la confiabilidad y la calidad del software.	Russo (2024)
Despliegue	Problemas de integración con sistemas existentes.	Falta de compatibilidad técnica.	Asegurar que las herramientas de AI sean compatibles con los sistemas actuales.	Reduce errores en producción y mejora la eficiencia del despliegue.	Russo (2024)
Mantenimiento	Preocupaciones sobre la seguridad y privacidad de los datos.	Entrenamiento de modelos en datos sensibles.	Utilizar datasets seguros y anonimizar datos para cumplir con regulaciones.	Incrementa la confianza en el uso de AI y asegura el cumplimiento normativo.	Russo (2024)
Planificación	Falta de confianza en herramientas de AI para	Dependencia de datos públicos para entrenar	Implementar políticas de gobernanza de datos y	Mejora la calidad de los requisitos y reduce	Shah & Trehan (2024)



	tareas críticas.	modelos.	validación manual.	errores iniciales.	
Diseño	Complejidad en la generación de prototipos innovadores.	Limitaciones en la creatividad autónoma de las herramientas.	Usar AI como complemento para acelerar la creación de prototipos.	Incrementa la velocidad en la fase de diseño sin comprometer la calidad.	Shah & Trehan (2024)
Desarrollo	Generación de código con errores o vulnerabilidades potenciales.	Dependencia excesiva en herramientas de IA.	Supervisión activa y revisión humana del código generado por IA.	Reduce errores y mejora la calidad del código final.	Shah & Trehan (2024)
Pruebas	Dificultades para detectar errores contextuales en el código generado.	Falta de validación exhaustiva en las herramientas de IA.	Combinar pruebas automatizadas con revisiones manuales para asegurar calidad.	Incrementa la confiabilidad y la robustez del software.	Shah & Trehan (2024)
Despliegue	Riesgos de seguridad en configuraciones automatizadas.	Falta de controles de seguridad en las herramientas de despliegue.	Implementar auditorías de seguridad y monitoreo continuo.	Mejora la seguridad y estabilidad en entornos de producción.	Shah & Trehan (2024)
Mantenimiento	Dificultades para gestionar soluciones generadas por AI a largo plazo.	Falta de documentación clara y reproducibilidad en el código generado.	Crear estándares de documentación y procesos de revisión periódica.	Asegura la sostenibilidad y facilita el soporte técnico.	Shah & Trehan (2024)
Planificación	Resistencia organizacional al cambio.	Temor al desplazamiento laboral y falta de comprensión	Implementar programas de capacitación y comunicación clara sobre los	Mejora la aceptación y facilita la integración de la AI en los	Al-kfairy (2025)



		de los beneficios de la IA.	beneficios de la IA.	procesos iniciales.	
Diseño	Complejidad en la integración de AI con sistemas heredados.	Sistemas antiguos no diseñados para interactuar con tecnologías modernas.	Desarrollar sistemas escalables y flexibles que permitan la integración gradual de IA.	Acelera los ciclos de diseño y mejora la adaptabilidad a nuevos requisitos.	Al-kfairy (2025)
Desarrollo	Falta de estándares establecidos para la implementaci ón de IA.	Naturaleza emergente de la tecnología y ausencia de guías claras.	Crear marcos de mejores prácticas y estándares para la integración de IA.	Reduce inconsistencia s y mejora la calidad del código generado.	Al-kfairy (2025)
Pruebas	Riesgos de sesgos en los datos utilizados para entrenar modelos de IA.	Datos históricos que reflejan desigualdades sistémicas.	Implementar algoritmos de aprendizaje justo y auditorías regulares de decisiones de IA.	Incrementa la confiabilidad y asegura resultados más equitativos.	Al-kfairy (2025)
Despliegue	Falta de transparencia en las decisiones tomadas por sistemas de IA.	Fenómeno de "caja negra" en modelos de aprendizaje profundo.	Desarrollar herramientas de explicabilidad para clarificar las decisiones de IA.	Mejora la confianza de los usuarios y facilita la adopción de la tecnología.	Al-kfairy (2025)
Mantenimie nto	Necesidad de actualizacione s continuas para mantener la relevancia de los modelos de IA.	Evolución constante de la tecnología y cambios en los contextos empresariales.	Establecer procesos de actualización periódica y monitoreo continuo de los sistemas de IA.	Asegura la sostenibilidad y la eficacia a largo plazo de las soluciones de IA.	Al-kfairy (2025)



Planificación	Incapacidad para traducir requisitos vagos en especificaciones técnicas claras.	Dependencia en entradas ambiguas de los stakeholders.	Uso de herramientas de AI con NLP para convertir lenguaje natural en requisitos técnicos.	Mejora la precisión en la definición de requisitos y evita problemas futuros.	Keskar (2023)
Diseño	Lentitud en la creación de arquitecturas y prototipos.	Proceso manual y subjetivo de diseño.	Uso de AI generativa para producir diagramas UML y wireframes de forma automática.	Agiliza los tiempos de diseño y asegura alineación funcional y técnica.	Keskar (2023)
Desarrollo	Generación de código con errores o patrones subóptimos.	Limitaciones humanas en identificar errores en tiempo real.	Implementación de herramientas como GitHub Copilot para generar y optimizar código.	Reduce errores y acelera el desarrollo con código estándar y de calidad.	Keskar (2023)
Pruebas	Cobertura limitada de casos de prueba y alta carga manual.	Enfoque tradicional basado en scripts predefinidos.	Uso de AI para generar matrices de prueba automatizadas y realizar depuración predictiva.	Mejora la cobertura de pruebas y detecta problemas antes de la producción.	Keskar (2023)
Despliegue	Problemas de configuración y riesgos en entornos de producción.	Falta de supervisión proactiva durante el despliegue.	Uso de AI para monitorear despliegues en tiempo real y resolver patrones críticos.	Incrementa la estabilidad y reduce fallas en despliegues complejos.	Keskar (2023)



Mantenimiento	Falta de proactividad en la identificación de fallos y la gestión de actualizaciones.	Monitoreo reactivo basado en intervenciones humanas.	Implementar análisis predictivo y generación automática de parches mediante IA.	Asegura la sostenibilidad del sistema y mejora la satisfacción del usuario final.	Keskar (2023)
Planificación	Ambigüedad en la definición de requisitos.	Dependencia en entradas humanas y falta de herramientas avanzadas de análisis.	Uso de AI con NLP para analizar documentos y estructurar requisitos automáticamente.	Mejora la precisión y reduce errores en etapas posteriores.	Shao et al. (2025)
Diseño	Complejidad en la creación de arquitecturas escalables.	Falta de herramientas para simulación y evaluación de diseños.	Implementar AI para generar propuestas de diseño y evaluar rendimiento.	Acelera la toma de decisiones y optimiza el diseño inicial.	Shao et al. (2025)
Desarrollo	Generación de código con errores y falta de robustez.	Limitaciones en la capacidad de los modelos actuales para manejar tareas complejas.	Uso de agentes de desarrollo para generar y revisar código automáticamente.	Incrementa la calidad del código y reduce el tiempo de desarrollo.	Shao et al. (2025)
Pruebas	Cobertura limitada de casos de prueba.	Enfoque manual y falta de automatización en la generación de pruebas.	Uso de AI para generar casos de prueba inteligentes y realizar diagnósticos automáticos.	Mejora la cobertura y detecta errores más rápidamente.	Shao et al. (2025)



Despliegue	Riesgos en la configuración y estabilidad del software en producción.	Falta de monitoreo proactivo y herramientas de diagnóstico.	Implementar AI para monitoreo en tiempo real y resolución de fallos.	Incrementa la estabilidad y reduce interrupciones en producción.	Shao et al. (2025)
Mantenimiento	Dificultades para actualizar y mantener sistemas complejos.	Falta de integración de conocimiento externo y herramientas predictivas.	Uso de agentes de gestión del conocimiento con RAG para actualizaciones dinámicas.	Asegura la sostenibilidad y mejora la eficiencia en el soporte técnico.	Shao et al. (2025)
Planificación	Ambigüedad en la definición de requisitos.	Dependencia de entradas humanas y falta de datos estructurados.	Usar NLP y machine learning para transformar lenguaje natural en especificaciones técnicas precisas.	Mejora la claridad en requisitos, reduciendo errores en etapas posteriores.	Oluwaseun (2023), pág. 594
Diseño	Dificultades para crear arquitecturas adaptativas.	Ausencia de herramientas avanzadas para simulación y evaluación de diseño.	Implementar herramientas de simulación impulsadas por AI que evalúen rendimiento en tiempo real.	Incrementa la velocidad de diseño y asegura estructuras más escalables.	Oluwaseun (2023), pág. 594
Desarrollo	Generación de código con errores debido a datos de baja calidad.	Falta de curación en los datasets utilizados para entrenar IA.	Desarrollar herramientas con algoritmos robustos que detecten y corrijan errores automáticamente.	Eleva la calidad del código, reduciendo el tiempo de depuración manual.	Oluwaseun (2023), pág. 595



Pruebas	Cobertura limitada y alta dependencia de pruebas manuales.	Enfoques tradicionales que no optimizan casos de prueba.	Uso de herramientas de prueba automatizadas basadas en AI para generación de test cases.	Mejora la cobertura y reduce tiempo y recursos necesarios en la fase de pruebas.	Oluwaseun (2023), pág. 596
Despliegue	Riesgos en entornos de producción por falta de monitoreo avanzado.	Falta de sistemas predictivos para monitoreo en tiempo real.	Implementar monitoreo predictivo con AI que detecte y mitigue fallos antes de que ocurran.	Incrementa la estabilidad y reduce interrupciones en entornos de producción.	Oluwaseun (2023), pág. 597
Mantenimiento	Falta de capacidad para prevenir problemas antes de que ocurran.	Ausencia de mantenimiento predictivo basado en análisis de datos históricos.	Uso de AI para mantenimiento predictivo que permita anticipar fallas en componentes críticos.	Extiende la vida útil del software y optimiza recursos técnicos a largo plazo.	Oluwaseun (2023), pág. 597



Anexo 3: Evaluación de la Calidad de los Artículos

Este anexo detalla el proceso de evaluación de la calidad de los estudios incluidos en esta revisión narrativa, siguiendo el enfoque propuesto por Kitchenham y otros autores en revisiones sistemáticas de ingeniería de software. El propósito de esta evaluación es asegurar que los artículos considerados aporten evidencia sólida. Se aplicaron criterios específicos que permiten valorar, de forma cuantitativa y cualitativa, aspectos como la claridad de los objetivos, la metodología utilizada, la reproducibilidad, la validez de los resultados y la relevancia de las contribuciones al tema de integración de la AI en el SDLC.

Criterios de Evaluación

La calidad de cada estudio se evaluó según los siguientes criterios:

1. Claridad en los Objetivos:

- **Descripción:** Se analizó si el artículo expone de manera explícita y precisa los objetivos de la investigación.
- **Importancia:** Un artículo con objetivos claramente definidos facilita la comprensión de la intención del estudio y la alineación con la pregunta de investigación.

2. Metodología Utilizada:

- **Descripción:** Se examinó la descripción del enfoque metodológico, los procedimientos de recolección y análisis de datos, y el uso de técnicas reconocidas (por ejemplo, la metodología de Kitchenham).
- **Importancia:** Una metodología bien descrita asegura que los hallazgos se basen en un proceso riguroso y replicable.

3. Reproducibilidad:

- **Descripción:** Se evaluó si el artículo proporciona suficientes detalles que permitan a otros investigadores replicar el estudio.
- **Importancia:** La reproducibilidad es clave para verificar y validar los resultados, lo que incrementa la fiabilidad de la investigación.



4. Validez de los Resultados:

- **Descripción:** Se analizaron los mecanismos utilizados para controlar variables, la validez interna y externa, y la fiabilidad de los resultados (por ejemplo, el uso de pruebas estadísticas o validaciones cualitativas).
- **Importancia:** Los resultados válidos indican que las conclusiones se sustentan en datos y análisis consistentes, lo cual es indispensable para la construcción de un marco teórico sólido.

5. Relevancia y Contribución:

- **Descripción:** Se consideró en qué medida el estudio aporta nuevo conocimiento o se diferencia de investigaciones previas, así como su aplicabilidad en la práctica profesional.
- **Importancia:** Estudios altamente relevantes refuerzan la fundamentación teórica y posibilitan la formulación de estrategias prácticas para la integración de la AI en el SDLC.

Cada artículo fue puntuado de 1 a 5 en cada uno de estos aspectos, donde 1 indica un desempeño muy deficiente y 5 representa un desempeño excelente. La suma total de estos puntajes proporciona una visión cuantitativa de la calidad general de cada estudio.

Proceso de Evaluación

El proceso seguido fue el siguiente:

1. **Selección Inicial:** Se identificaron y descargaron los artículos relevantes a través de Google Scholar, complementados con búsquedas en otras bases de datos reconocidas.
2. **Filtrado y Clasificación:** Se revisaron títulos y resúmenes para descartar estudios que no cumplieran con los criterios de inclusión definidos previamente. Los artículos seleccionados se organizaron en una hoja de cálculo.
3. **Aplicación de la Matriz de Evaluación:** Cada artículo se evaluó utilizando la matriz de calidad, asignándole puntajes en los criterios mencionados (Claridad en Objetivos,



Metodología, Reproducibilidad, Validez y Relevancia). De esta forma, se obtuvo una puntuación total que permitió distinguir los estudios de mayor rigor y contribución.

Presentación de la Matriz

La siguiente tabla ilustra el formato de la matriz de evaluación utilizada:

Artículo	Claridad en Objetivos	Metodología	Reproducibilidad	Validez	Relevancia	Total
A1	5	4	3	5	4	21
A2	4	3	2	4	5	18

Este anexo facilita la apreciación del nivel de calidad de cada uno de los estudios incluidos en la revisión, permitiendo que los hallazgos y conclusiones se basen en información sólida y confiable. La metodología de evaluación adoptada, en combinación con la matriz de evaluación, permite discernir cómo cada artículo contribuye a la comprensión de los desafíos y barreras en la integración de la AI en el SDLC, y fundamenta las estrategias y recomendaciones presentadas en el trabajo.

Tabla 3. Evaluación de Calidad

Artículo	Claridad en Objetivos (1-5)	Metodología (1-5)	Reproducibilidad (1-5)	Validez (1-5)	Relevancia (1-5)	Puntuación Total
Durrani, U. K., Akpinar, M., Adak, M. F., Kabakus, A. T., Ozturk, M. M., & Saleh, M. (2024). A decade of progress: A systematic literature review on the integration of ai in software engineering phases and activities (2013-2023). IEEE Access.	El artículo presenta claramente su objetivo de analizar la integración de AI en las fases del SDLC. (5)	El artículo utiliza una revisión sistemática de la literatura adoptando el método de Kitchenham para analizar tendencias y desafíos. (5)	Describe detalladamente la metodología, incluyendo las técnicas de revisión y las bases de datos utilizadas. (5)	Los resultados están respaldados por una revisión exhaustiva de literatura con referencias relevantes. (4)	Aborda directamente la integración de AI en la ingeniería de software, siendo muy relevante al campo. (5)	24/25
KAMBALA, G. (2024). Intelligent Software Agents for Continuous Delivery: Leveraging AI and Machine Learning for Fully Automated DevOps Pipelines.	El artículo tiene como objetivo principal explicar cómo agentes inteligentes y aprendizaje automático se aplican para automatizar pipelines de DevOps. (5)	Utiliza un enfoque descriptivo con ejemplos prácticos para explicar las aplicaciones de los agentes inteligentes en DevOps. (4)	Aunque se describen casos prácticos, la metodología carece de suficientes detalles técnicos para una completa reproducibilidad. (3)	Las aplicaciones propuestas están respaldadas por ejemplos reales y discusiones sobre su implementación en DevOps. (4)	Aborda la integración de AI y agentes inteligentes en DevOps, un área crucial en ingeniería de software. (5)	21/25

**“Desde los Obstáculos hasta las Estrategias: Integración
de la inteligencia artificial en el ciclo de vida del
desarrollo del software.”**

Giffari, R., Ridho, M. M., Senseuse, D. I., Hidayat, D. S., & Purwaningsih, E. H. (2024). Analyst's Perception on the Use of AI-based Tools in the Software Development Life Cycle. <i>Jurnal Sistem Informasi</i> , 20(1), 73-87.	El artículo tiene como objetivo explorar las percepciones de los analistas sobre el uso de herramientas basadas en AI en la fase de análisis del SDLC. (5)	Utiliza un enfoque cualitativo con entrevistas semiestructuradas y análisis temático para recopilar y analizar datos. (5)	Aunque describe el proceso de entrevistas y análisis, no incluye detalles suficientes para replicar exactamente los resultados. (3)	los resultados están respaldados por las percepciones de los analistas entrevistados y referencias a estudios previos relevantes. (4)	Aborda la integración de herramientas de AI en el SDLC, un tema clave en la ingeniería de software. (5)	22/25
Benitez, C. D., & Serrano, M. (2023). The Integration and Impact of Artificial Intelligence in Software Engineering. <i>Integration</i> , 3(2).	El artículo establece claramente su objetivo de analizar la integración y el impacto de la AI en la ingeniería de software. (5)	Utiliza un enfoque sistemático para analizar las aplicaciones de AI en las diferentes fases del SDLC, respaldado por ejemplos y referencias. (5)	Aunque describe las aplicaciones y beneficios de la IA, no incluye detalles técnicos suficientes para una completa reproducibilidad. (3)	Las conclusiones están respaldadas por estudios previos y ejemplos prácticos de integración de AI en el SDLC. (4)	Aborda directamente la integración de AI en la ingeniería de software, un tema clave en el campo. (5)	22/25
Khan, A. R., Fatima, J., & Ahmed, S. (2023). Crossover of Artificial Intelligence and Software Development Lifecycle. <i>International Journal of Computing and Related Technologies</i> , 4(1), 35-42.	El artículo establece claramente su objetivo de analizar la intersección entre AI y SDLC, identificando	Utiliza un enfoque sistemático basado en literatura revisada y ejemplos prácticos para respaldar sus hallazgos. (5)	Describe los métodos y herramientas utilizadas, pero carece de detalles técnicos para una completa	Las conclusiones están respaldadas por estudios previos y análisis detallados de casos prácticos. (4)	Aborda un tema clave en la integración de AI en el SDLC, destacando desafíos y estrategias. (5)	22/25

	desafíos y barreras. (5)		reproducibilidad. (3)			
Garg, K. (2023). Impact of Artificial Intelligence on software development: Challenges and Opportunities. <i>International Journal of Software & Hardware Research in Engineering</i> , 11(8).	El artículo establece claramente su objetivo de analizar el impacto de la AI en el desarrollo de software, destacando desafíos y oportunidades. (5)	Utiliza un enfoque descriptivo basado en análisis de casos prácticos y tendencias de la industria. (4)	Describe los desafíos y oportunidades, pero carece de detalles técnicos para una completa reproducibilidad. (3)	Las conclusiones están respaldadas por ejemplos reales y referencias a estudios previos relevantes. (4)	Aborda un tema crucial en la integración de AI en el desarrollo de software. (5)	21/25
Kumar, S. (2024). Artificial Intelligence in Software Engineering: A Systematic Exploration of AI-Driven Development.	El artículo establece claramente su objetivo de explorar sistemáticamente el impacto de la AI en el desarrollo de software. (5)	Utiliza un enfoque sistemático basado en revisiones de literatura y análisis de casos prácticos. (4)	Describe los métodos y herramientas utilizadas, pero carece de detalles técnicos para una completa reproducibilidad. (3)	Las conclusiones están respaldadas por ejemplos prácticos y referencias a estudios previos relevantes. (4)	Aborda un tema crucial en la integración de AI en el desarrollo de software. (5)	21/25

**“Desde los Obstáculos hasta las Estrategias: Integración
de la inteligencia artificial en el ciclo de vida del
desarrollo del software.”**

Müller, M. (2024). Advancements in Software Engineering through Artificial Intelligence.	El artículo establece objetivos claros, enfocados en el impacto y las aplicaciones de la AI en ingeniería de software. (5)	Utiliza un enfoque mixto con análisis comparativo y estudios de caso, especialmente en el sector de la salud. (4)	Describe resultados y herramientas, pero faltan detalles técnicos para una reproducibilidad completa. (3)	Están respaldados por casos prácticos y análisis cuantitativos significativos. (4)	Trata desafíos clave y propone estrategias directamente aplicables al desarrollo de software. (5)	21/25
Haghsheno, S. (2020). <i>Revolutionizing Software Engineering: Leveraging AI for Enhanced Development Lifecycle. International Journal of Innovative Research in Modern Physics and Software (IJIRMPs)</i> , 8(1), January-February.	El artículo establece claramente su propósito de explorar el impacto de la AI en el ciclo de vida del desarrollo de software.(5)	Utiliza un enfoque descriptivo y analítico, apoyado en ejemplos prácticos y estudios de caso. (4)	Describe aplicaciones y desafíos, pero carece de detalles técnicos para una completa reproducibilidad. (3)	Las conclusiones están respaldadas por ejemplos prácticos y análisis detallados de aplicaciones de IA. (4)	Aborda desafíos clave y propone estrategias directamente aplicables al desarrollo de software. (5)	21/25
Ajiga, D., Okeleke, P. A., Folorunsho, S. O., & Ezeigweneme, C. (2024). Enhancing software development practices with AI insights in high-tech companies. <i>IEEE Software Engineering Institute, Technical Report TR-2024-003</i> .	El artículo establece claramente su propósito de explorar cómo la AI mejora las prácticas de desarrollo de software en empresas de alta	Utiliza un enfoque descriptivo y analítico, apoyado en estudios de caso y ejemplos prácticos. (4)	Describe aplicaciones y desafíos, pero carece de detalles técnicos para una completa reproducibilidad. (3)	Las conclusiones están respaldadas por ejemplos prácticos y análisis detallados de aplicaciones de IA. (4)	Aborda desafíos clave y propone estrategias directamente aplicables al desarrollo de software. (5)	21/25

	tecnología. (5)					
Zhang, Q. (2024). The Role of Artificial Intelligence in Modern Software Engineering. <i>Applied and Computational Engineering</i> , 97, 18-23.	El artículo establece claramente el propósito de explorar cómo la AI influye en todas las etapas del ciclo de vida del desarrollo de software. (5)	El artículo usa un enfoque descriptivo y analiza aplicaciones prácticas, pero carece de detalles experimentales que permitan reproducir los resultados. (3)	El artículo describe herramientas y aplicaciones, pero no detalla los algoritmos o modelos utilizados para su evaluación. (3)	Las conclusiones están bien fundamentadas y respaldadas por ejemplos concretos de herramientas como Codex y Test.AI. (4)	Aborda desafíos clave y propone soluciones directamente aplicables a la integración de AI en software. (5)	20/25
Siddique, I. (2022). Harnessing artificial intelligence for systems engineering: Promises and pitfalls. <i>European Journal of Advances in Engineering and Technology</i> , 9(9), 67-72.	Los objetivos son claros, centrándose en explorar las promesas y dificultades de la AI en la ingeniería de sistemas. Sin embargo, no abarca todos los aspectos del ciclo de vida del	El artículo utiliza estudios de caso y análisis cualitativo, pero carece de experimentos prácticos o comparativos que permitan validar plenamente los resultados. (3)	Se presentan ejemplos y herramientas, pero no se incluye suficiente detalle técnico para replicar los estudios o experimentos en otros contextos.	Las conclusiones son coherentes y basadas en análisis, pero algunas generalizaciones carecen de evidencia cuantitativa robusta. (4)	aborda desafíos importantes relacionados con la integración de AI en sistemas de ingeniería y su impacto en el ciclo de vida del software. (5)	19/25

	software. (4)		(3)			
Madupati, B. (2025). AI's Impact on Traditional Software Development. <i>arXiv preprint arXiv:2502.18476</i> .	El artículo establece objetivos claros, como analizar el impacto de la AI en metodologías tradicionales de desarrollo de software. Sin embargo, podría profundizar más en algunos aspectos específicos del SDLC. (4)	Utiliza un enfoque descriptivo y analítico, apoyado en estudios de caso y ejemplos prácticos. Sin embargo, carece de experimentos empíricos que validen los resultados. (3)	Aunque se describen herramientas y aplicaciones, no se incluyen detalles técnicos suficientes para replicar los resultados en otros contextos. (3)	Las conclusiones están bien fundamentadas en análisis y ejemplos prácticos, pero algunas generalizaciones carecen de datos cuantitativos robustos. (4)	Aborda desafíos clave y propone soluciones directamente aplicables a la integración de AI en el desarrollo de software. (5)	19/25

**“Desde los Obstáculos hasta las Estrategias: Integración
de la inteligencia artificial en el ciclo de vida del
desarrollo del software.”**

Alenezi, M., & Akour, M. (2025). AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions. <i>Applied Sciences</i> , 15(3), 1344.	Los objetivos están claramente definidos, centrándose en revisar prácticas actuales y explorar direcciones futuras en la integración de AI en ingeniería de software. (5)	El artículo utiliza un enfoque descriptivo y analítico, apoyado en estudios de caso y revisión de literatura relevante. Sin embargo, carece de experimentos empíricos. (4)	Aunque se describen herramientas y aplicaciones, no se incluyen detalles técnicos suficientes para replicar los resultados en otros contextos. (3)	Las conclusiones están bien fundamentadas en análisis y ejemplos prácticos, pero algunas generalizaciones carecen de datos cuantitativos robustos. (4)	Aborda desafíos clave y propone soluciones directamente aplicables a la integración de AI en el desarrollo de software. (5)	21/25
Vadde, B. C., & Munagandla, V. B. (2022). AI-Driven Automation in DevOps: Enhancing Continuous Integration and Deployment. <i>International Journal of Advanced Engineering Technologies and Innovations</i> , 1(3), 183-193.	los objetivos están claramente definidos, centrándose en cómo la AI mejora la automatización en DevOps, especialmente en CI/CD. (5)	el artículo utiliza un enfoque descriptivo y analítico, apoyado en estudios de caso y ejemplos prácticos. Sin embargo, carece de experimentos empíricos. (4)	Aunque se describen herramientas y aplicaciones, no se incluyen detalles técnicos suficientes para replicar los resultados en otros contextos. (3)	Las conclusiones están bien fundamentadas en análisis y ejemplos prácticos, pero algunas generalizaciones carecen de datos cuantitativos robustos. (4)	Aborda desafíos clave y propone soluciones directamente aplicables a la integración de AI en DevOps. (5)	21/25

Upadhyaya, N. (2022). <i>The Role of Artificial Intelligence in Software Development: A Literature Review</i> . ResearchGate.	Los objetivos están claramente definidos, centrándose en el impacto de la AI en todas las etapas del ciclo de vida del desarrollo de software. (5)	El artículo utiliza una revisión exhaustiva de literatura y ejemplos prácticos, pero carece de validación empírica o experimentos. (4)	Aunque se describen herramientas y aplicaciones, faltan detalles técnicos para replicar los resultados en otros contextos. (3)	Las conclusiones están bien fundamentadas en análisis y ejemplos prácticos, pero algunas generalizaciones carecen de datos cuantitativos robustos. (4)	Aborda desafíos clave y propone soluciones directamente aplicables a la integración de AI en el desarrollo de software. (5)	21/25
Kokol, P. (2024). The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature. <i>Information</i> , 15(6), 354.	Los objetivos están claramente definidos, centrándose en sintetizar el conocimiento reciente sobre el uso de AI en ingeniería de software. (5)	El artículo utiliza una síntesis de conocimiento basada en una revisión sistemática de literatura reciente, pero carece de validación empírica. (4)	Aunque se describen herramientas y técnicas, faltan detalles técnicos para replicar los resultados en otros contextos. (3)	Las conclusiones están bien fundamentadas en análisis de literatura, pero algunas generalizaciones carecen de datos cuantitativos robustos. (4)	Aborda desafíos clave y propone soluciones directamente aplicables a la integración de AI en el desarrollo de software. (5)	21/25

Sergeyuk, A., Zakharov, I., Koshchenko, E., & Izadi, M. (2025). Human-AI Experience in Integrated Development Environments: A Systematic Literature Review. <i>arXiv preprint arXiv:2503.06195</i> .	Los objetivos están claramente definidos, centrándose en la experiencia humano-IA en entornos de desarrollo integrados. (5)	El artículo utiliza una revisión sistemática de literatura, pero carece de validación empírica o estudios experimentales. (4)	Aunque se describen herramientas y técnicas, faltan detalles técnicos para replicar los resultados en otros contextos. (3)	Las conclusiones están bien fundamentadas en análisis de literatura, pero algunas generalizaciones carecen de datos cuantitativos robustos. (4)	Aborda desafíos clave y propone soluciones directamente aplicables a la integración de AI en el desarrollo de software. (5)	21/25
Özpolat, Z., Yıldırım, Ö., & Karabatak, M. (2023). Artificial intelligence-based tools in software development processes: Application of chatgpt. <i>European Journal of Technique (EJT)</i> , 13(2), 229-240.	Los objetivos están claramente definidos, centrándose en el impacto de herramientas basadas en AI como ChatGPT en el desarrollo de software. (5)	El artículo utiliza un enfoque sistemático para analizar aplicaciones de IA, pero carece de validación empírica en algunos casos. (4)	Aunque se describen herramientas y técnicas, faltan detalles técnicos para replicar los resultados en otros contextos. (3)	Las conclusiones están bien fundamentadas en análisis de literatura y ejemplos prácticos, pero algunas generalizaciones carecen de datos cuantitativos robustos. (4)	Aborda desafíos clave y propone soluciones directamente aplicables a la integración de AI en el desarrollo de software. (5)	21/25

**“Desde los Obstáculos hasta las Estrategias: Integración
de la inteligencia artificial en el ciclo de vida del
desarrollo del software.”**

Medepalli, S. (2025). AI In Modern Software Development: Current Practices, Challenges and Future Possibilities. <i>Journal Publication of International Research for Engineering and Management (JOIREM)</i> , 10(01).	Los objetivos están claramente definidos, centrándose en los desafíos y oportunidades de la AI en el desarrollo de software. (5)	El artículo utiliza un enfoque sistemático basado en entrevistas con expertos y análisis de literatura, aunque carece de validación empírica. (4)	Aunque se describen herramientas y técnicas, faltan detalles técnicos para replicar los resultados en otros contextos. (3)	Las conclusiones están bien fundamentadas en análisis cualitativos, pero algunas generalizaciones carecen de datos cuantitativos robustos. (4)	Aborda desafíos clave y propone soluciones directamente aplicables a la integración de AI en el desarrollo de software. (5)	21/25
Todorov, P. G. (2022). The Application of Artificial Intelligence in Software Engineering. <i>Available at SSRN</i> .	El artículo define claramente sus objetivos, enfocándose en cuatro áreas clave de la integración de AI en ingeniería de software. (5)	Presenta una revisión bibliográfica y un análisis de aplicaciones prácticas, aunque falta detalle en cómo se implementaron las técnicas. (4)	Las técnicas y modelos mencionados son replicables, pero la falta de especificación técnica completa puede limitar su implementación. (4)	Las conclusiones están bien respaldadas por referencias relevantes, aunque se podría profundizar en casos prácticos específicos. (4)	El tema tratado es altamente relevante y actual, abordando desafíos y oportunidades cruciales en el uso de AI en ingeniería de software. (5)	22/25

**“Desde los Obstáculos hasta las Estrategias: Integración
de la inteligencia artificial en el ciclo de vida del
desarrollo del software.”**

Pothukuchi, A. S., Kota, L. V., & Mallikarjunaradhya, V. (2023). Impact of generative ai on the software development lifecycle (sdlc). <i>International Journal of Creative Research Thoughts</i> , 11(8).	El artículo define claramente su objetivo principal: analizar cómo la AI generativa impacta cada fase del SDLC y proponer un modelo adaptado (GAASD). (5)	Utiliza un enfoque interpretativo con entrevistas y encuestas a profesionales de la industria, aunque el tamaño de la muestra es limitado (30 participantes). (4)	Aunque describe el proceso de entrevistas y análisis, no proporciona detalles suficientes para replicar completamente el estudio. (3)	Los resultados están respaldados por datos empíricos y revisiones externas, pero la generalización está limitada por el tamaño de la muestra. (4)	El tema es altamente relevante, abordando un área crítica en la evolución del desarrollo de software con AI generativa. (5)	21/25
Ricárdez, J. J. U., Vasconcelos, M. P., Domínguez, R. G., & Romero, F. C. (2024). Integration of Generative AI with ChatGPT in Software Development.	El artículo presenta claramente los objetivos de integrar AI generativa en el desarrollo de software, pero podría mejorar en la especificidad de los problemas abordados. (4)	Se describe detalladamente la metodología, incluyendo herramientas, frameworks y modelos de desarrollo como Agile Scrum. (5)	Aunque se presentan casos de uso y ejemplos, algunos detalles específicos de implementación no están completamente desarrollados. (4)	Se apoya en estudios y análisis recientes, incluyendo tendencias y predicciones sobre el impacto de la AI en la industria. (5)	Dada la creciente adopción de AI en el desarrollo de software, el tema es altamente pertinente y útil para la comunidad de desarrolladores. (5)	23/25

Upadhyaya, N. AI-Powered Secure Software Development: The Future of Safe and Efficient Coding.	El artículo define claramente su objetivo principal: analizar el impacto de la AI generativa en el desarrollo seguro de software y proponer estrategias para mitigar vulnerabilidades. (5)	Utiliza un enfoque basado en estudios previos y análisis de herramientas como GitHub Copilot, aunque carece de experimentación empírica directa. (4)	Aunque describe casos prácticos y ejemplos, no proporciona detalles suficientes para replicar completamente el análisis realizado. (3)	Los resultados están respaldados por referencias relevantes y estudios previos, pero la generalización está limitada por la dependencia de datos públicos. (4)	El tema tratado es altamente relevante, abordando un área crítica en la evolución del desarrollo de software seguro con AI generativa. (5)	21/25
Russo, D. (2024). Navigating the complexity of generative ai adoption in software engineering. <i>ACM Transactions on Software Engineering and Methodology</i> , 33(5), 1-50.	El artículo define claramente su objetivo principal: explorar los factores que influyen en la adopción de herramientas de AI generativa en ingeniería de software, utilizando un marco teórico robusto (HACAF). (5)	Utiliza un enfoque mixto convergente, combinando métodos cualitativos y cuantitativos, con validación mediante PLS-SEM, lo que garantiza rigor y profundidad en el análisis. (5)	Aunque detalla los pasos de la investigación, como el diseño de encuestas y la metodología Gioia, algunos aspectos del análisis cualitativo podrían ser difíciles de replicar sin acceso a los datos originales. (4)	Los resultados están respaldados por datos empíricos de 183 ingenieros de software y análisis estadísticos robustos, lo que refuerza la validez del modelo HACAF. (5)	El tema tratado es altamente relevante, abordando un área crítica en la integración de AI generativa en el desarrollo de software, con implicaciones prácticas y teóricas significativas. (5)	24/25

Shah, K., & Trehan, A. (2024). Streamlining software development: A comparative study of AI-driven automation tools in modern tech. <i>International Journal of Computer Engineering and Technology</i>, 15(6), 1638-1650.	El artículo define claramente su objetivo principal: analizar herramientas de automatización impulsadas por AI en el desarrollo de software, destacando sus ventajas y limitaciones. (5)	Utiliza un enfoque descriptivo y comparativo, combinando análisis cualitativo y cuantitativo. Sin embargo, podría incluir más detalles sobre la selección de herramientas y datos. (4)	Aunque describe las herramientas y su impacto, no proporciona datos completos o accesibles para replicar el análisis en otros contextos. (4)	Los resultados están respaldados por un análisis exhaustivo de herramientas populares como UiPath, Automation Anywhere y GitHub Copilot, lo que refuerza la validez de las conclusiones. (5)	El tema es altamente relevante, abordando desafíos actuales en la integración de AI en el ciclo de vida del desarrollo de software, con implicaciones prácticas significativas. (5)	23/25
Al-kfairy, M. (2025). Strategic Integration of Generative AI in Organizational Settings: Applications, Challenges and Adoption Requirements. <i>IEEE Engineering Management Review</i>.	El artículo define claramente sus objetivos, centrándose en las aplicaciones, desafíos y requisitos estratégicos para la adopción de AI generativa en organizaciones. (5)	Utiliza una revisión sistemática de la literatura con criterios rigurosos de selección y evaluación de calidad, lo que asegura un análisis exhaustivo y confiable. (5)	Aunque detalla el proceso de selección y análisis, la falta de acceso a algunos datos originales podría dificultar la replicación exacta del estudio. (4)	Los resultados están respaldados por un análisis temático sólido y referencias a 38 estudios de alta calidad, lo que refuerza la validez de las conclusiones. (5)	El tema es altamente relevante, abordando un área crítica en la integración de AI generativa en entornos organizacionales, con implicaciones prácticas significativas. (25)	24/25

Keskar, A. (2023). Revolutionizing Software Engineering with Generative AI: Transforming Development Processes, Automation, and Quality Assurance. <i>TIJER - International Research Journal</i> , 10(6), 830–844.	El artículo define claramente sus objetivos, centrándose en cómo la AI generativa transforma los procesos de desarrollo, automatización y aseguramiento de calidad en ingeniería de software. (5)	Utiliza un enfoque descriptivo y casos prácticos para respaldar sus afirmaciones, aunque podría incluir más detalles sobre la selección de herramientas y datos. (4)	Aunque describe herramientas y ejemplos, no proporciona datos completos para replicar el análisis en otros contextos. (4)	Los resultados están respaldados por un análisis exhaustivo de herramientas como GitHub Copilot y Tabnine, lo que refuerza la validez de las conclusiones. (5)	El tema es altamente relevante, abordando desafíos actuales en la integración de AI generativa en el ciclo de vida del desarrollo de software, con implicaciones prácticas significativas. (5)	23/25
Shao, X., Shibasaki, M., & Shibasaki, R. (2025, February). Artificial Intelligence in Software Engineering: Integration and Key Challenges. In <i>CS & IT Conference Proceedings</i> (Vol. 15, No. 4). CS & IT Conference Proceedings.	El artículo define claramente sus objetivos, centrándose en los desafíos y oportunidades de integrar AI en el ciclo de vida del desarrollo de software. (5)	Utiliza un enfoque descriptivo y casos prácticos para respaldar sus afirmaciones, aunque podría incluir más detalles sobre la implementación de herramientas específicas. (4)	Aunque describe herramientas y ejemplos, no proporciona datos completos para replicar el análisis en otros contextos. (4)	Los resultados están respaldados por un análisis exhaustivo de herramientas como MetaGPT y RAG, lo que refuerza la validez de las conclusiones. (5)	El tema es altamente relevante, abordando desafíos actuales en la integración de AI en el ciclo de vida del desarrollo de software, con implicaciones prácticas significativas. (5)	23/25

Oluwaseun, A. W. (2023). Harnessing AI for Advanced Software Engineering: Opportunities and Challenges. <i>Journal of Emerging Technologies and Innovative Research</i> , 10(5), 594–599.	El artículo define claramente sus objetivos, centrándose en cómo la AI puede transformar el ciclo de vida del desarrollo de software, destacando aplicaciones, desafíos y oportunidades. (5)	Utiliza un enfoque mixto (cualitativo y cuantitativo) que incluye revisión sistemática de literatura, análisis bibliométrico y estudios de caso, aunque podría profundizar más en los criterios de selección de datos. (4)	Proporciona detalles sobre las herramientas y técnicas utilizadas, pero carece de datos completos para replicar los análisis en otros contextos. (4)	Los resultados están respaldados por un análisis exhaustivo de aplicaciones de AI en software y casos prácticos en otros campos como la salud, lo que refuerza la validez de las conclusiones. (5)	El tema es altamente relevante, abordando desafíos actuales en la integración de AI en el desarrollo de software, con implicaciones prácticas significativas para la industria. (5)	23/25
--	--	--	--	--	---	-------