Tipo de Artículo: *Artículo de revisión*
Área: *Ingeniería*

Patrones Arquitectónicos en la Generación de Código Asistida por IA: Revisión Sistemática

Architectural Patterns in AI-Assisted Code Generation: A Systematic Review

Jaime David Camacho Castillo, Christian Fernando Barragán Quizhpe, Alex Javier Canchignia Vasco, Enrique Marcelo Baño León, Santiago Enrique Paltan García

Cita:
Camacho Castillo, J. D., Barragán Quizhpe, C. F., Canchignia Vasco, A. J., Baño León, E. M., & Paltan García, S. E. (2026). Patrones arquitectónicos en la generación de código asistida por IA: Revisión sistemática. TESLA Revista Científica, 6(2), e691. https://doi.org/10.55204/trc.v6i2.e691

Recibido: 2026-06-08
Revisado: 2026-06-10 al 2026-06-19
Corregido: 2026-06-29
Aceptado: 2026-07-01
Publicado: 2026-07-03

1 Escuela Superior Politécnica de Chimborazo (ESPOCH); 2 Universidad Estatal de Bolívar UEB; 3 Universidad de las Fuerzas Armadas (ESPE); 4-5 Investigador Externo.

1 jaimed.camacho@espoch.edu.ec; 2 cbarragan@ueb.edu.ec; 3 ajcanchignia@espe.edu.ec; 4 enrique.bano@ueb.edu.ec; 5 segarcia@bce.ec

1 https://orcid.org/0000-0002-9110-6585; 2 https://orcid.org/0000-0003-4699-9553; 3 https://orcid.org/0009-0003-5904-3806; 4 https://orcid.org/0000-0002-5550-0649; 5 https://orcid.org/0009-0002-1428-6719

Resumen: La arquitectura de software desempeña un papel fundamental en el desarrollo de sistemas informáticos complejos, ya que define decisiones estructurales de alto nivel que influyen directamente en atributos de calidad como la mantenibilidad, la escalabilidad y la confiabilidad. De forma paralela, los avances recientes en inteligencia artificial generativa, en particular en modelos de lenguaje de gran escala, han impulsado su adopción en distintas fases de la ingeniería de software, principalmente en la generación automática de código. No obstante, los enfoques actuales tienden a centrarse en el nivel de implementación, prestando atención limitada a las decisiones arquitectónicas y al uso explícito de patrones arquitectónicos. Este trabajo presenta una revisión estructurada del estado del arte sobre la aplicación de inteligencia artificial generativa en la arquitectura de software, con énfasis en la relación entre requisitos, decisiones arquitectónicas y generación de código asistida por IA. La metodología se basa en un análisis cualitativo y comparativo de literatura científica reciente, identificando enfoques, beneficios y limitaciones reportadas. Los resultados evidencian que la IA generativa puede apoyar el diseño arquitectónico temprano, la alineación arquitectura código y el análisis arquitectónico, aunque persisten desafíos como la falta de datasets arquitectónicos, la limitada interpretabilidad y la ausencia de mecanismos de evaluación sistemática. Se concluye que la integración explícita de patrones arquitectónicos como conocimiento estructurado es clave para mejorar la coherencia y calidad del desarrollo de software asistido por inteligencia artificial.

Palabras clave: *Arquitectura de software, Patrones arquitectónicos, Inteligencia artificial generativa, Generación de código asistida por IA, Ingeniería de software.*

Abstract: Software architecture plays a fundamental role in the development of complex computing systems, as it defines high-level structural decisions that directly influence quality attributes such as maintainability, scalability, and reliability. In parallel, recent advancements in generative artificial intelligence, particularly in large language models, have driven its adoption across different phases of software engineering, primarily in automatic code generation. However, current approaches tend to focus on the implementation level, paying limited attention to architectural decisions and the explicit use of architectural patterns. This work presents a structured review of the state of the art regarding the application of generative artificial intelligence in software architecture, emphasizing the relationship between requirements, architectural decisions, and AI-assisted code generation. The methodology is based on a qualitative and comparative analysis of recent scientific literature, identifying approaches, benefits, and reported limitations. The results show that generative AI can support early architectural design, architecture-code alignment, and architectural analysis, although challenges persist, such as the lack of architectural datasets, limited interpretability, and the absence of systematic evaluation mechanisms. It is concluded that the explicit integration of architectural patterns as structured knowledge is key to improving the consistency and quality of software development assisted by artificial intelligence.

Keywords: *Health in older adults, Nutritional assessment, Web application, XP methodology, Software efficiency.*

INTRODUCCIÓN

La arquitectura de software constituye uno de los pilares fundamentales en el desarrollo de sistemas informáticos complejos, ya que proporciona una visión de alto nivel que orienta la organización de los componentes, la asignación de responsabilidades y la toma de decisiones técnicas a lo largo del ciclo de vida del software. En este contexto, los patrones arquitectónicos han surgido como soluciones recurrentes a problemas comunes de diseño, permitiendo mejorar atributos de calidad como la mantenibilidad, escalabilidad, reutilización y robustez de los sistemas. Tradicionalmente, la selección y aplicación de estos patrones ha dependido en gran medida de la experiencia humana, del conocimiento especializado de los arquitectos de software y del análisis manual de los requisitos del sistema.

Paralelamente, el avance acelerado de la inteligencia artificial (IA), y en particular de la IA generativa basada en modelos de lenguaje de gran escala, ha comenzado a transformar de manera significativa diversas

fases del ciclo de vida del software. Estudios recientes evidencian un creciente interés en el uso de estas tecnologías para apoyar actividades como la elicitación y análisis de requisitos, la generación de código, la validación de artefactos y la toma de decisiones de diseño (Cheng et al., 2026). Los sistemas de IA generativa han demostrado capacidades notables para interpretar descripciones en lenguaje natural, sintetizar información técnica y producir código funcional a partir de especificaciones de alto nivel, lo que ha impulsado su adopción tanto en entornos académicos como industriales.

Uno de los ámbitos donde la influencia de la IA generativa resulta especialmente relevante es la generación de código asistida por IA. En este escenario, los desarrolladores interactúan con herramientas inteligentes capaces de sugerir estructuras de proyecto, generar fragmentos de código y proponer soluciones técnicas de forma automatizada. No obstante, a pesar de estos avances, la mayoría de las soluciones actuales se enfocan principalmente en el nivel de implementación, prestando una atención limitada a las decisiones arquitectónicas que subyacen al software generado. Esta desconexión entre la generación automática de código y la arquitectura de software puede derivar en sistemas inconsistentes, difíciles de mantener o poco alineados con los requisitos no funcionales.

Diversas investigaciones han señalado que la falta de una integración explícita entre la IA generativa y los principios arquitectónicos limita el impacto positivo de estas tecnologías en la ingeniería de software (Cheng et al., 2026). En particular, se ha identificado una brecha en la traducción sistemática de los requisitos hacia decisiones arquitectónicas coherentes, especialmente en sistemas modernos caracterizados por arquitecturas distribuidas, microservicios, entornos en la nube y altas exigencias de calidad. Esta problemática pone en evidencia la necesidad de analizar cómo los patrones arquitectónicos están siendo considerados, explícita o implícitamente, en los procesos de generación de código asistida por IA.

Desde una perspectiva más amplia, estudios recientes destacan la importancia de incorporar el contexto del dominio, las restricciones del entorno y las necesidades del usuario para lograr soluciones de software más adaptativas y sostenibles (Liu et al., 2026). En este sentido, la IA generativa no debería limitarse a producir código sintácticamente correcto, sino que debería ser capaz de razonar sobre decisiones arquitectónicas, patrones aplicables y atributos de calidad esperados. Sin embargo, la literatura existente muestra que los análisis sistemáticos sobre esta integración aún son limitados, ya que la mayoría de los trabajos se concentran en aspectos algorítmicos o en la evaluación del desempeño de los modelos, dejando de lado una visión arquitectónica integral.

Ante este escenario, el objetivo de este artículo de revisión es analizar de manera sistemática el estado del arte sobre la relación entre los patrones arquitectónicos y la generación de código asistida por inteligencia artificial, con el fin de identificar enfoques existentes, beneficios, limitaciones y brechas de investigación. La importancia de este estudio radica en la necesidad de proporcionar un marco conceptual que permita comprender cómo la arquitectura de software puede integrarse de forma explícita en los procesos asistidos por IA, contribuyendo a mejorar la calidad, sostenibilidad y escalabilidad del software generado automáticamente. En consecuencia, el problema central abordado en este trabajo es la falta de consenso y sistematización sobre el rol de los patrones arquitectónicos en la generación de código asistida por IA, cuestión que esta revisión busca esclarecer a partir del análisis de la literatura científica reciente.

METODOLOGÍA

Tipo de Estudio

La presente investigación se desarrolló mediante una revisión sistemática de la literatura (Systematic Literature Review, SLR), con un enfoque cualitativo y comparativo. Este tipo de estudio permite analizar, sintetizar y evaluar de manera estructurada la evidencia científica existente sobre un tema específico, garantizando rigor metodológico y reproducibilidad en los resultados obtenidos.

La revisión se centró en estudios recientes relacionados con la arquitectura de software, los patrones arquitectónicos y el uso de inteligencia artificial generativa en la generación de código, con el objetivo de identificar tendencias, enfoques predominantes y vacíos de investigación.

Metodología PRISMA

Para asegurar un proceso de revisión riguroso, transparente y trazable, se adoptó la metodología PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses). PRISMA es un marco metodológico ampliamente reconocido que proporciona directrices claras para la identificación, selección, evaluación y

síntesis de estudios científicos, reduciendo sesgos y mejorando la calidad de las revisiones sistemáticas.

La versión actualizada de PRISMA establece criterios estandarizados para documentar cada fase del proceso de revisión, permitiendo una evaluación clara del impacto y relevancia de los estudios seleccionados (Fuentes-Quijada et al., 2025).

Aplicación del Método PRISMA en la Investigación

La aplicación del método PRISMA en esta investigación siguió prácticas recomendadas en revisiones sistemáticas dentro del ámbito de la ingeniería de software. Estudios recientes demuestran que PRISMA es especialmente adecuado para analizar literatura técnica compleja, permitiendo estructurar investigaciones relacionadas con procesos, herramientas y enfoques emergentes en ingeniería de software y arquitectura de sistemas (Rodrigues et al., 2025).

El proceso seguido incluyó las siguientes etapas:

- **Identificación:** Búsqueda de estudios en bases de datos científicas indexadas, empleando palabras clave relacionadas con arquitectura de software, patrones arquitectónicos, inteligencia artificial generativa y generación de código asistida por IA.
- **Cribado:** Eliminación de duplicados y revisión preliminar de títulos y resúmenes para descartar estudios no alineados con el objetivo de la investigación.
- **Elegibilidad:** Evaluación del texto completo de los estudios seleccionados, considerando criterios de calidad, relevancia temática y actualidad.
- **Inclusión:** Selección final de los estudios que cumplieron con todos los criterios definidos, conformando el conjunto de referencias analizadas en profundidad.

Justificación del uso de la Metodología PRISMA

La elección de la metodología PRISMA se justifica por su amplia adopción en estudios de ingeniería de software y arquitectura de software, donde ha demostrado ser efectiva para estructurar revisiones sistemáticas y mapear el estado del arte de manera objetiva. Investigaciones previas han utilizado PRISMA para alinear conceptos de ingeniería de software y gestión de productos, evidenciando su aplicabilidad en contextos arquitectónicos y organizacionales (Oruthotaarachchi & Wijayanayake, 2025).

A continuación, se presentan las principales razones que fundamentan el uso de PRISMA en esta investigación:

Tabla 1

Justificación del uso de la metodología PRISMA

Criterio	Justificación
Rigor metodológico	Garantiza un proceso de revisión estructurado, sistemático y reproducible
Evaluación del impacto	Permite priorizar estudios relevantes según criterios de calidad y actualidad
Transparencia	Documenta de forma clara cada etapa del proceso de selección de estudios
Reducción de sesgos	Minimiza la selección subjetiva de literatura mediante criterios definidos
Alineación académica	Es un método estándar ampliamente aceptado en ingeniería de software

Fuente: Elaboración propia.

Resultados del proceso de selección de estudios

Como resultado de la aplicación de la metodología PRISMA, se obtuvo un conjunto final de estudios relevantes que sustentan el análisis presentado en este trabajo. A continuación, se resumen las principales etapas del proceso de selección:

Tabla 2

Resultados del proceso de selección de estudios mediante PRISMA

Etapa del proceso	Cantidad de estudios
Estudios identificados	48
Estudios tras eliminación de duplicados	30
Estudios evaluados por título y resumen	32
Estudios evaluados en texto completo	22
Estudios incluidos en la revisión	20

Fuente: Elaboración propia.

ARQUITECTURA DE SOFTWARE Y PATRONES ARQUITECTÓNICOS

La arquitectura de software constituye un elemento central en el desarrollo de sistemas complejos, ya que define las estructuras fundamentales del sistema, las decisiones de diseño clave y la relación entre los distintos artefactos que intervienen en el ciclo de vida del software. En esta sección se revisan los fundamentos de la arquitectura de software, el rol de los patrones arquitectónicos en los atributos de calidad, las arquitecturas modernas basadas en microservicios y eventos, y el papel de la arquitectura como puente entre los requisitos y la implementación.

Fundamentos de la arquitectura de software

La arquitectura de software puede entenderse como el conjunto de decisiones estructurales que determinan la organización de un sistema y su evolución a lo largo del tiempo. Estas decisiones no solo afectan la estructura estática del sistema, sino también su comportamiento, su capacidad de adaptación y la trazabilidad entre los distintos artefactos de desarrollo.

Un aspecto clave de la arquitectura es la gestión explícita de las decisiones arquitectónicas y su relación con los artefactos de software. El estudio de (Hyun & Hurtado, 2023) analiza de forma sistemática la trazabilidad entre las decisiones de diseño arquitectónico y los artefactos asociados, evidenciando que la falta de trazabilidad dificulta el mantenimiento, la evolución y la comprensión del sistema. Los autores destacan que una arquitectura bien documentada y trazable facilita la evaluación de impactos ante cambios en los requisitos y mejora la calidad global del sistema.

Patrones arquitectónicos y atributos de calidad

Los patrones arquitectónicos proporcionan soluciones reutilizables a problemas recurrentes en el diseño de sistemas de software. Su adopción influye directamente en atributos de calidad como la mantenibilidad, la escalabilidad, la modularidad y la resiliencia.

El trabajo presentado por (Nouman et al., 2023) examina la relación entre los patrones arquitectónicos y los atributos de calidad del software, destacando que la selección adecuada de un patrón no es neutra, sino que favorece ciertos atributos mientras compromete otros. Este enfoque resulta especialmente relevante para el análisis comparativo de patrones arquitectónicos, ya que permite justificar su uso en función de objetivos de calidad específicos, en lugar de decisiones puramente tecnológicas.

Arquitecturas modernas basadas en microservicios y eventos

Las arquitecturas modernas han evolucionado hacia enfoques distribuidos, siendo los microservicios y las arquitecturas orientadas a eventos dos de los estilos más representativos. Estas arquitecturas buscan desacoplar componentes, aumentar la autonomía de los equipos y facilitar la escalabilidad del sistema.

El estudio empírico presentado por (Dhawan, 2026) analiza una arquitectura basada en microservicios habilitada por un gateway con soporte WebSocket, evidenciando cómo el desacoplamiento entre la velocidad de despliegue y la gobernanza de la plataforma permite mejorar la agilidad sin sacrificar el control arquitectónico. Los resultados muestran que las decisiones arquitectónicas en este tipo de sistemas tienen un impacto directo en la escalabilidad, el rendimiento y la capacidad de evolución del software.

La arquitectura como puente entre requisitos , diseño e implementación

La arquitectura de software cumple un rol fundamental como elemento de transición entre los requisitos del sistema y su implementación concreta. En este sentido, actúa como un mecanismo de traducción que permite pasar de modelos conceptuales y decisiones de alto nivel a estructuras implementables.

El trabajo presentado por (Le et al., 2025) propone un enfoque basado en arquitecturas modulares y orientadas al dominio que conecta explícitamente los requisitos con el diseño arquitectónico y el código resultante. Este enfoque refuerza la idea de la arquitectura como un puente entre los distintos niveles de abstracción del desarrollo de software, facilitando la automatización parcial, la coherencia del diseño y la evolución controlada del sistema.

DESARROLLO

Arquitectura de Software y Patrones Arquitectónicos

La arquitectura de software constituye una disciplina fundamental dentro de la ingeniería de software, orientada a la definición de la estructura de alto nivel de un sistema y a la organización de sus componentes, relaciones y principios de diseño. Desde una perspectiva conceptual, la arquitectura permite establecer una visión global del sistema que trasciende los detalles de implementación, facilitando la toma de decisiones estratégicas relacionadas con la evolución, el mantenimiento y la calidad del software. En sistemas complejos y de gran escala, la arquitectura se convierte en un elemento crítico para gestionar la complejidad, reducir riesgos técnicos y garantizar la alineación entre los requisitos del negocio y las soluciones tecnológicas.

En este contexto, los patrones arquitectónicos surgen como soluciones reutilizables a problemas recurrentes de diseño a nivel estructural. Un patrón arquitectónico define un esquema organizacional probado que describe cómo se distribuyen las responsabilidades del sistema, cómo interactúan sus componentes y qué restricciones deben cumplirse para alcanzar determinados atributos de calidad. A diferencia de los patrones de diseño, que operan a un nivel más cercano a la implementación, los patrones arquitectónicos influyen directamente en la forma global del sistema y en su comportamiento a largo plazo.

La adopción de patrones arquitectónicos aporta múltiples beneficios al desarrollo de software. Entre los más relevantes se encuentran la mejora de la mantenibilidad, la escalabilidad, la reutilización de soluciones, la interoperabilidad y la robustez del sistema. Asimismo, los patrones permiten capturar conocimiento experto acumulado a lo largo del tiempo, facilitando que arquitectos y desarrolladores tomen decisiones informadas sin partir desde cero en cada nuevo proyecto. Este aspecto resulta especialmente relevante en entornos donde los sistemas deben evolucionar rápidamente o adaptarse a cambios constantes en los requisitos funcionales y no funcionales.

Diversos estilos arquitectónicos han sido ampliamente utilizados en la práctica y documentados en la literatura. Entre los más representativos se encuentran la arquitectura en capas, el estilo cliente–servidor, los sistemas monolíticos, la arquitectura orientada a servicios (SOA) y, más recientemente, la arquitectura de microservicios. Cada uno de estos estilos responde a necesidades específicas y presenta ventajas y limitaciones en función del contexto de aplicación. Por ejemplo, mientras las arquitecturas monolíticas pueden simplificar el desarrollo inicial, las arquitecturas basadas en microservicios favorecen la escalabilidad y la independencia de despliegue, a costa de una mayor complejidad operativa.

La selección de un patrón arquitectónico adecuado está estrechamente relacionada con los atributos de calidad que se desean priorizar en el sistema. Atributos como rendimiento, escalabilidad, seguridad, disponibilidad y tolerancia a fallos dependen en gran medida de las decisiones arquitectónicas adoptadas en las primeras fases del desarrollo. En consecuencia, una arquitectura mal diseñada puede comprometer seriamente la calidad del sistema, incluso si el código fuente cumple con los requisitos funcionales especificados. Por esta razón, la arquitectura de software es considerada un factor determinante en el éxito o fracaso de proyectos complejos.

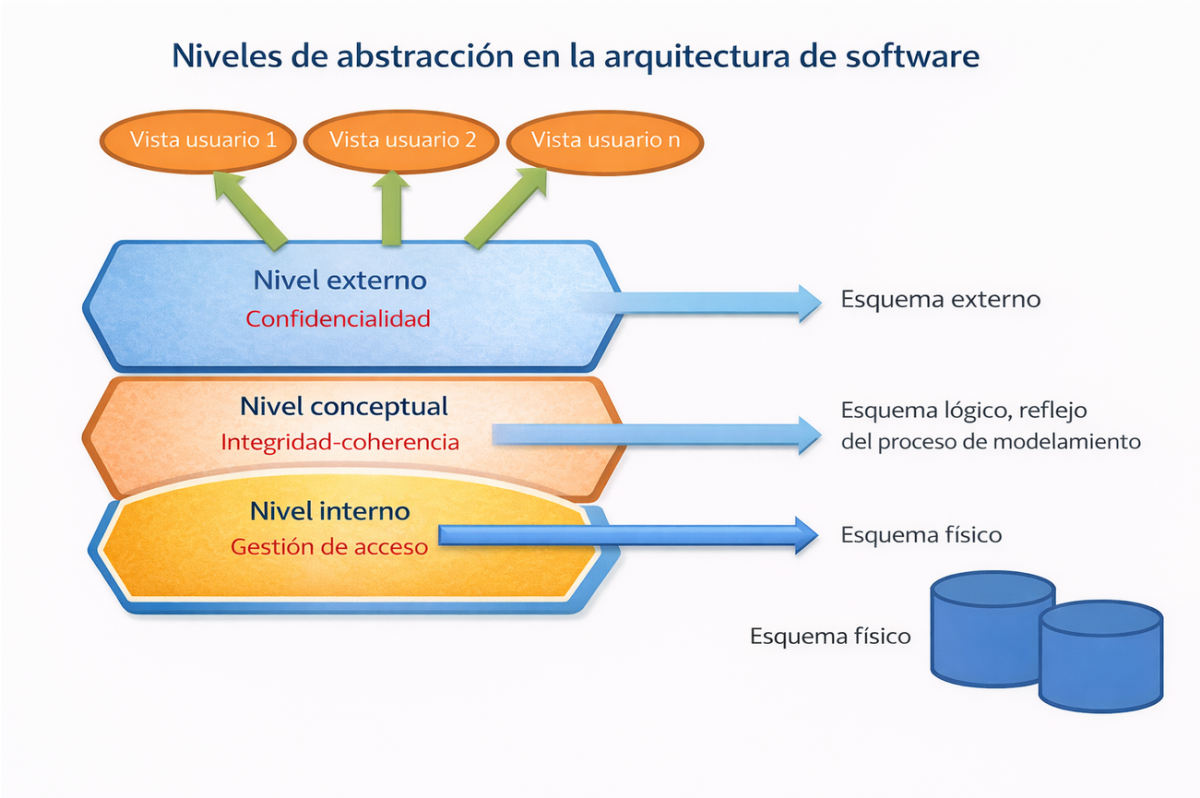
En los últimos años, la creciente adopción de tecnologías emergentes, como la computación en la nube, los sistemas distribuidos y las aplicaciones basadas en inteligencia artificial, ha incrementado la relevancia de los patrones arquitectónicos. Estos entornos introducen nuevos desafíos relacionados con la gestión de datos, la comunicación entre componentes, la latencia y la escalabilidad dinámica, lo que exige arquitecturas más flexibles y adaptativas. En este escenario, los patrones arquitectónicos continúan desempeñando un papel clave como mecanismos para estructurar sistemas complejos y responder a las demandas actuales del desarrollo de software.

Desde una perspectiva teórica, la arquitectura de software actúa como un puente entre los requisitos del sistema y su implementación técnica. Las decisiones arquitectónicas traducen necesidades abstractas en estructuras concretas que guían el desarrollo posterior del código. Este vínculo resulta particularmente

relevante cuando se analizan enfoques automatizados o asistidos por inteligencia artificial, ya que pone de manifiesto la necesidad de preservar la coherencia arquitectónica incluso cuando parte del proceso de desarrollo se delega a sistemas inteligentes. En consecuencia, comprender los fundamentos de la arquitectura de software y los patrones arquitectónicos constituye un paso indispensable para analizar su integración con tecnologías de generación de código asistida por inteligencia artificial, tema que se aborda en las subsecciones siguientes.

Figura 1

Niveles de abstracción en la arquitectura de software



Fuente: Elaboración propia.

Como se observa en la ***Figura 1*** , la arquitectura de software se estructura en distintos niveles de abstracción que permiten separar las preocupaciones del sistema. El nivel externo se orienta a las vistas de usuario y a aspectos como la confidencialidad; el nivel conceptual representa el esquema lógico resultante del proceso de modelado, garantizando coherencia e integridad; mientras que el nivel interno describe el esquema físico y los mecanismos de gestión de acceso. Esta separación facilita el control de la complejidad y la evolución del sistema sin afectar de manera directa a las demás capas.

Tabla 3

Relación entre patrones arquitectónicos y atributos de calidad

Patrón arquitectónico	Atributos de calidad y consideraciones
Arquitectura en capas	Favorece la mantenibilidad y la separación de responsabilidades; puede introducir sobrecarga en el rendimiento si no se diseña adecuadamente.
Cliente-servidor	Permite centralizar servicios y facilitar la administración; la escalabilidad y la disponibilidad dependen de mecanismos adicionales como balanceo de carga y replicación.
Microservicios	Promueve la escalabilidad, el despliegue independiente y la evolución modular; incrementa la complejidad operativa y la necesidad de herramientas DevOps.
Arquitectura orientada a eventos	Favorece el desacoplamiento y la alta reactividad; puede dificultar la trazabilidad y la consistencia de datos si no se gestiona correctamente.
Pipe and Filter	Facilita la reutilización y el procesamiento secuencial de datos; puede introducir latencia en flujos extensos o distribuidos.
Broker (intermediación)	Mejora la interoperabilidad entre componentes heterogéneos; puede convertirse en un punto crítico si no se dimensiona adecuadamente.

Fuente: Elaboración propia.

Inteligencia Artificial Generativa en la Ingeniería de Software

IA generativa en el ciclo de vida del software

Dentro de la ingeniería de software, la IA generativa ha sido aplicada en múltiples fases del proceso de desarrollo. En la etapa de ingeniería de requisitos, estos modelos se utilizan para apoyar la elicitación, análisis, clasificación y validación de requisitos, permitiendo transformar descripciones informales en especificaciones más estructuradas. Estudios recientes destacan que la IA generativa puede asistir en la identificación de ambigüedades, inconsistencias y dependencias entre requisitos, mejorando la calidad de los artefactos producidos en etapas tempranas del desarrollo (Al-Azzoni et al., 2026), (Esposito et al., 2026a).

Asimismo, en fases posteriores del ciclo de vida del software, la IA generativa ha mostrado utilidad en la generación automática de código, la sugerencia de fragmentos de implementación, la creación de pruebas unitarias y la generación de documentación técnica. Estas capacidades permiten reducir el esfuerzo manual, acelerar los ciclos de desarrollo y facilitar la adopción de prácticas ágiles, especialmente en entornos donde la presión por reducir el tiempo de entrega es elevada (Esposito et al., 2026a) .

No obstante, la literatura enfatiza que la efectividad de la IA generativa a lo largo del ciclo de vida del software depende en gran medida de su correcta integración con procesos y prácticas consolidadas de la ingeniería de software. La ausencia de mecanismos que garanticen la trazabilidad entre requisitos, diseño y código puede limitar los beneficios de la automatización y generar artefactos difíciles de mantener o evolucionar a largo plazo (Al-Azzoni et al., 2026).

Modelos de lenguaje de gran escala en el desarrollo de software

Los modelos de lenguaje de gran escala constituyen el núcleo de muchas soluciones actuales de inteligencia artificial generativa aplicadas al desarrollo de software. Estos modelos, entrenados sobre grandes repositorios de código y documentación técnica, han demostrado capacidades avanzadas para generar código fuente, sugerir refactorizaciones y asistir en tareas de depuración. Investigaciones recientes reportan que los LLMs pueden mejorar la productividad de los desarrolladores, especialmente en tareas repetitivas o de bajo nivel (Abrahão et al., 2025).

Sin embargo, la literatura también advierte que los LLMs carecen de una comprensión explícita de la semántica arquitectónica y de los principios de diseño subyacentes al sistema. Como resultado, el código generado puede ser funcional, pero no necesariamente coherente con las decisiones arquitectónicas o los atributos de calidad esperados. Esta limitación resulta especialmente crítica en sistemas complejos y de gran escala, donde la arquitectura de software desempeña un rol determinante en la calidad global del sistema (Esposito et al., 2026a).

Desde esta perspectiva, diversos autores coinciden en que los LLMs deben concebirse como herramientas de apoyo al desarrollador y no como sustitutos del razonamiento humano. Su uso efectivo requiere la incorporación de contexto estructurado, restricciones explícitas y validación por parte de expertos, con el fin de mitigar riesgos asociados a la generación automática de soluciones subóptimas (Abrahão et al., 2025).

Riesgos y desafíos de la IA generativa en la ingeniería de software

A pesar de los beneficios identificados, la adopción de la inteligencia artificial generativa en la ingeniería de software plantea desafíos significativos. Uno de los principales riesgos señalados en la literatura es la dependencia del contexto: los modelos generativos tienden a producir resultados más precisos cuando disponen de información suficiente sobre el dominio, los requisitos y las restricciones del sistema. En ausencia de dicho contexto, los artefactos generados pueden resultar inconsistentes, incompletos o incorrectos (Al-Azzoni et al., 2026), (Esposito et al., 2026a) .

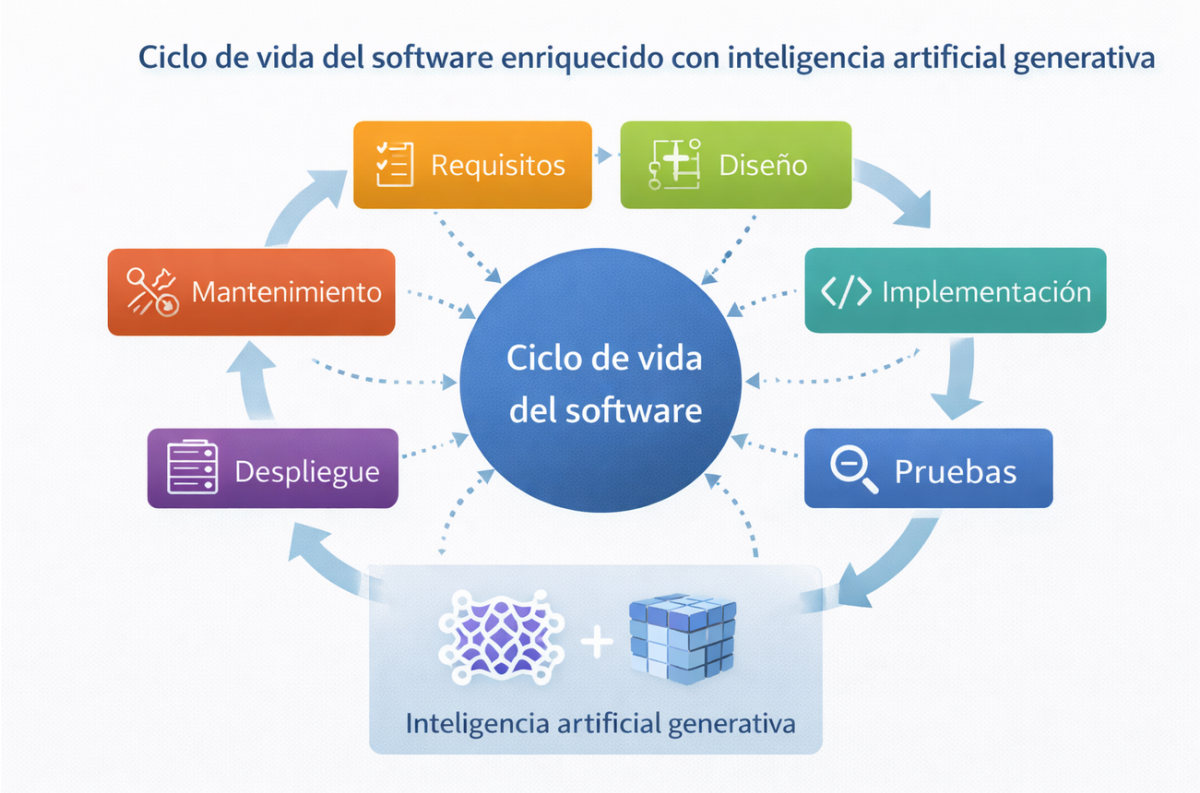
Otro desafío relevante se relaciona con la explicabilidad y la confianza en las decisiones generadas por los modelos. Debido a su naturaleza opaca, los LLMs dificultan la justificación de las soluciones propuestas, lo que complica su adopción en entornos críticos o regulados. Además, se ha reportado el riesgo de propagación de errores y malas prácticas de diseño, especialmente cuando el código generado se reutiliza sin una revisión exhaustiva (Abrahão et al., 2025).

Desde una perspectiva metodológica, la literatura destaca que la integración de la IA generativa en el ciclo de vida del software suele concentrarse en etapas cercanas a la implementación, dejando en un segundo plano las fases de diseño y arquitectura. Esta situación puede generar una desconexión entre requisitos, decisiones arquitectónicas y código, afectando atributos de calidad como la mantenibilidad, la escalabilidad y la coherencia estructural del sistema (Esposito et al., 2026a)

La ***Figura 2*** ilustra un ciclo de vida del software enriquecido con el uso de inteligencia artificial generativa. En este enfoque, la IA no actúa únicamente como una herramienta de generación de código, sino como un asistente transversal que apoya distintas etapas del proceso, desde la comprensión de requisitos hasta la producción de artefactos técnicos. Este modelo resalta la necesidad de una integración más sistemática de la IA generativa en el ciclo de vida completo, incorporando mecanismos que permitan preservar la trazabilidad y la coherencia entre las distintas fases del desarrollo (Esposito et al., 2026b).

Figura 2

Ciclo de vida del software enriquecido con inteligencia artificial generativa



Fuente: Elaboración propia.

En síntesis, la inteligencia artificial generativa representa una oportunidad significativa para transformar la ingeniería de software, siempre que su adopción se articule de manera coherente con prácticas consolidadas como la arquitectura de software y el diseño estructurado. La comprensión de su rol dentro del ciclo de

vida del software constituye un paso esencial para analizar, en secciones posteriores, cómo estas tecnologías pueden influir de forma más directa en decisiones arquitectónicas y en la generación de código alineada con patrones arquitectónicos bien definidos.

IA Generativa aplicada a la Arquitectura de Software

La aplicación de la inteligencia artificial generativa a la arquitectura de software constituye una evolución natural del uso de estas tecnologías dentro de la ingeniería de software. Mientras que los primeros enfoques de IA generativa se centraron en la generación automática de código, investigaciones recientes evidencian un interés creciente por extender su alcance hacia decisiones de diseño de mayor nivel, particularmente aquellas relacionadas con la arquitectura del sistema (Esposito et al., 2026a). Este cambio de enfoque responde a la necesidad de preservar la coherencia estructural del sistema y garantizar atributos de calidad desde las fases tempranas del desarrollo.

IA generativa y decisiones arquitectónicas

Uno de los principales aportes de la IA generativa en el ámbito arquitectónico es su capacidad para asistir en la toma de decisiones arquitectónicas complejas. A partir del análisis de requisitos, restricciones del dominio y experiencias previas codificadas en grandes volúmenes de datos, los modelos de lenguaje pueden sugerir alternativas arquitectónicas y estilos estructurales adecuados. No obstante, estudios recientes destacan que estas decisiones deben ser interpretables y justificables, dado que la falta de explicabilidad puede limitar su adopción en entornos industriales (Gacitúa et al., 2026).

Desde esta perspectiva, la IA generativa no reemplaza al arquitecto de software, sino que actúa como un asistente cognitivo que apoya el razonamiento arquitectónico humano. La literatura enfatiza que la supervisión humana sigue siendo indispensable para validar las decisiones propuestas y evaluar sus implicaciones sobre atributos de calidad como mantenibilidad, escalabilidad y seguridad (Gacitúa et al., 2026).

Enfoques Requirements-to-Architecture y Architecture-to-Code

Uno de los escenarios de aplicación más relevantes identificados en la literatura es el enfoque Requirements-to-Architecture. En este contexto, los modelos de lenguaje de gran escala son utilizados para analizar requisitos funcionales y no funcionales expresados en lenguaje natural, con el objetivo de inferir decisiones arquitectónicas iniciales. Estas decisiones pueden incluir la selección de estilos arquitectónicos, la identificación de componentes principales y la consideración de atributos de calidad como escalabilidad, mantenibilidad o seguridad (Esposito et al., 2026a). Este enfoque resulta especialmente relevante en fases tempranas del desarrollo, donde la arquitectura actúa como un elemento estructurante que condiciona el resto del proceso de construcción del software.

Un segundo enfoque ampliamente discutido es el denominado Architecture-to-Code, en el cual la arquitectura previamente definida se emplea como guía para la generación automática o semiautomática de código. En este caso, la IA generativa traduce modelos arquitectónicos o descripciones estructurales de alto nivel en artefactos de implementación, tales como esqueletos de proyectos, módulos de software o configuraciones iniciales. Investigaciones recientes proponen marcos unificados que integran requisitos, arquitectura y generación automática de artefactos, destacando el potencial de estos enfoques para mejorar la alineación entre diseño y código (Nguyen et al., 2026). Sin embargo, también se señala el riesgo de inconsistencias estructurales cuando las decisiones generadas no son validadas por expertos humanos.

Análisis y evaluación arquitectónica asistida por IA

Además de la generación de arquitecturas iniciales, la IA generativa ha sido aplicada en tareas de análisis y evaluación arquitectónica. En estos enfoques, los modelos generativos se utilizan para inferir arquitecturas a partir de sistemas existentes, analizar alternativas de diseño y evaluar soluciones arquitectónicas en función de atributos de calidad específicos. Este tipo de aplicaciones busca apoyar a los arquitectos de software en actividades de revisión y análisis, sin reemplazar su rol, actuando como herramientas de soporte para la detección temprana de problemas estructurales (Autili et al., 2026) código, investigaciones recientes evidencian un interés creciente por extender su alcance hacia decisiones de diseño de mayor nivel, particularmente aquellas relacionadas con la arquitectura del sistema (Esposito et al., 2026b).

Uno de los principales escenarios de aplicación identificados en la literatura es el enfoque *Requirements-to-Architecture*. En este contexto, los modelos de lenguaje de gran escala son utilizados para analizar re-

quisitos funcionales y no funcionales expresados en lenguaje natural, con el objetivo de inferir decisiones arquitectónicas iniciales. Estas decisiones pueden incluir la selección de estilos arquitectónicos, la identificación de componentes principales y la consideración de atributos de calidad como escalabilidad, mantenibilidad o seguridad. Este enfoque resulta especialmente relevante en fases tempranas del desarrollo, donde la arquitectura actúa como un elemento estructurante que condiciona el resto del proceso de construcción del software (Esposito et al., 2026b).

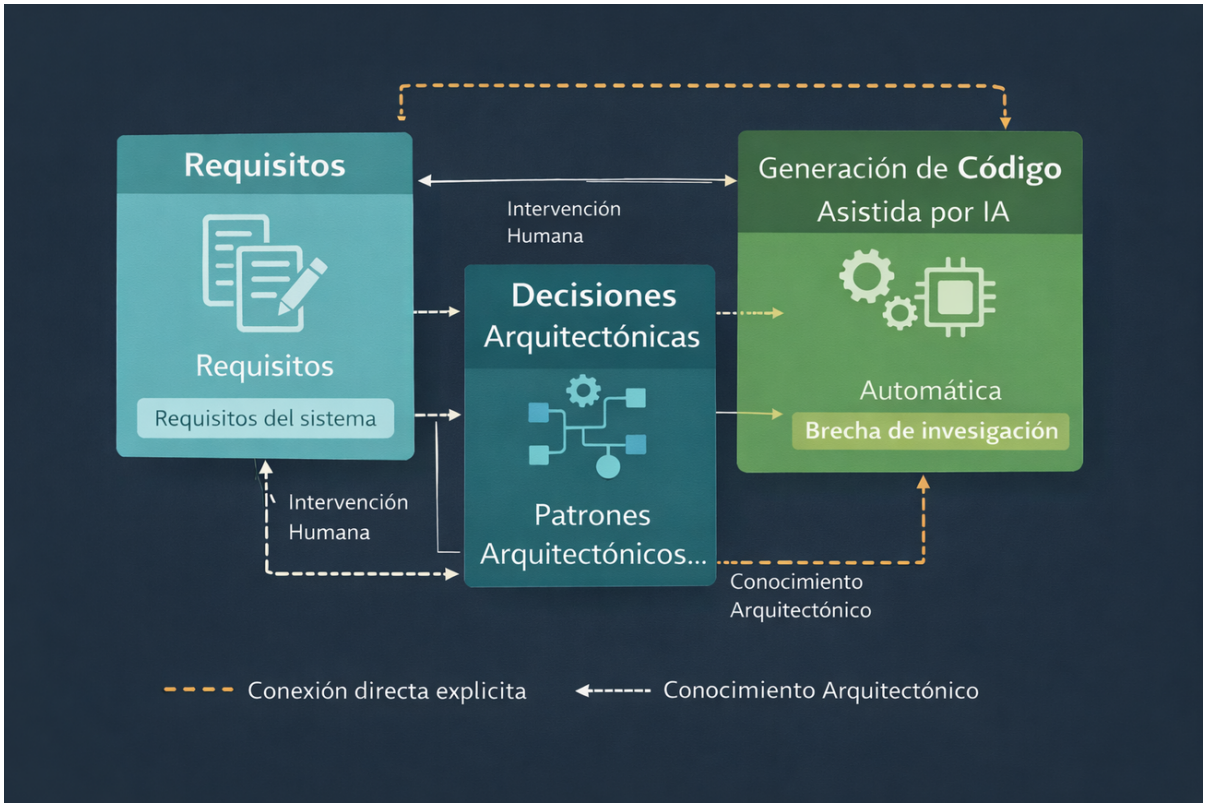
Un segundo enfoque relevante es el denominado *Architecture-to-Code*, en el cual la arquitectura definida previamente se emplea como guía para la generación automática o semiautomática de código. En este caso, la IA generativa traduce modelos arquitectónicos o descripciones estructurales de alto nivel en artefactos de implementación, tales como esqueletos de proyectos, módulos de software o configuraciones iniciales. La literatura señala que este enfoque puede contribuir a una mayor coherencia entre la arquitectura y el código, aunque también introduce riesgos asociados a la pérdida de control arquitectónico cuando las decisiones generadas no son validadas por expertos humanos (Esposito et al., 2026b).

Adicionalmente, se han propuesto enfoques orientados a la reconstrucción y análisis arquitectónico asistido por IA, en los cuales los modelos generativos se utilizan para inferir arquitecturas a partir de sistemas existentes o para evaluar soluciones arquitectónicas en función de determinados atributos de calidad. Estos enfoques buscan apoyar a los arquitectos de software en tareas de revisión y análisis, sin reemplazar su rol, actuando como herramientas de soporte para la detección temprana de problemas estructurales.

La ***Figura 3*** representa un modelo conceptual del flujo que conecta los requisitos, la arquitectura y el código en procesos asistidos por inteligencia artificial generativa. En este modelo, la arquitectura de software desempeña un rol central como elemento de enlace entre las necesidades del sistema y su implementación técnica, permitiendo que la IA generativa opere de manera más coherente y controlada.

Figura 3

Modelo conceptual Requisitos → Arquitectura → Código en procesos asistidos por IA generativa.



Fuente: Elaboración propia.

Desde una perspectiva comparativa, los principales enfoques de aplicación de la IA generativa en la arquitectura de software presentan beneficios y desafíos diferenciados, como se resume en la ***Tabla 4***. Esta síntesis permite evidenciar que, si bien la IA generativa puede acelerar y apoyar el proceso de diseño arquitectónico, su uso efectivo requiere mecanismos de validación, supervisión humana y conocimiento ar-

quitectónico explícito para evitar decisiones subóptimas.

Tabla 4

Enfoques de inteligencia artificial generativa aplicados a la arquitectura de software

Enfoque	Beneficios	Desafíos
Requirements-to-Architecture	Apoyo a la toma de decisiones arquitectónicas tempranas y consideración de atributos de calidad	Ambigüedad en los requisitos y dependencia del contexto
Architecture-to-Code	Alineación inicial entre arquitectura y código generado automáticamente	Riesgo de inconsistencias estructurales y pérdida de control arquitectónico
Análisis arquitectónico asistido	Detección temprana de problemas de diseño y apoyo a arquitectos de software	Limitada interpretabilidad de los modelos y falta de métricas de evaluación

Fuente: Elaboración propia.

Los enfoques revisados muestran que la inteligencia artificial generativa tiene el potencial de influir significativamente en la arquitectura de software, siempre que se utilice como un mecanismo de apoyo y no como un sustituto del razonamiento arquitectónico humano. La literatura coincide en que la arquitectura continúa siendo un factor crítico para la calidad del sistema, incluso en contextos donde la generación de código es asistida por IA, lo que refuerza la necesidad de integrar explícitamente principios y patrones arquitectónicos en estos procesos

Patrones Arquitectónicos en Sistemas Basados en Inteligencia Artifical

El crecimiento de los sistemas basados en inteligencia artificial ha puesto de manifiesto la necesidad de arquitecturas de software robustas que permitan gestionar la complejidad inherente a estos entornos. A diferencia de las aplicaciones tradicionales, los sistemas de IA incorporan componentes adicionales relacionados con la gestión de datos, el entrenamiento de modelos, la inferencia y la actualización continua del conocimiento, lo que incrementa la relevancia de las decisiones arquitectónicas adoptadas durante su diseño (Compagnucci et al., 2026).

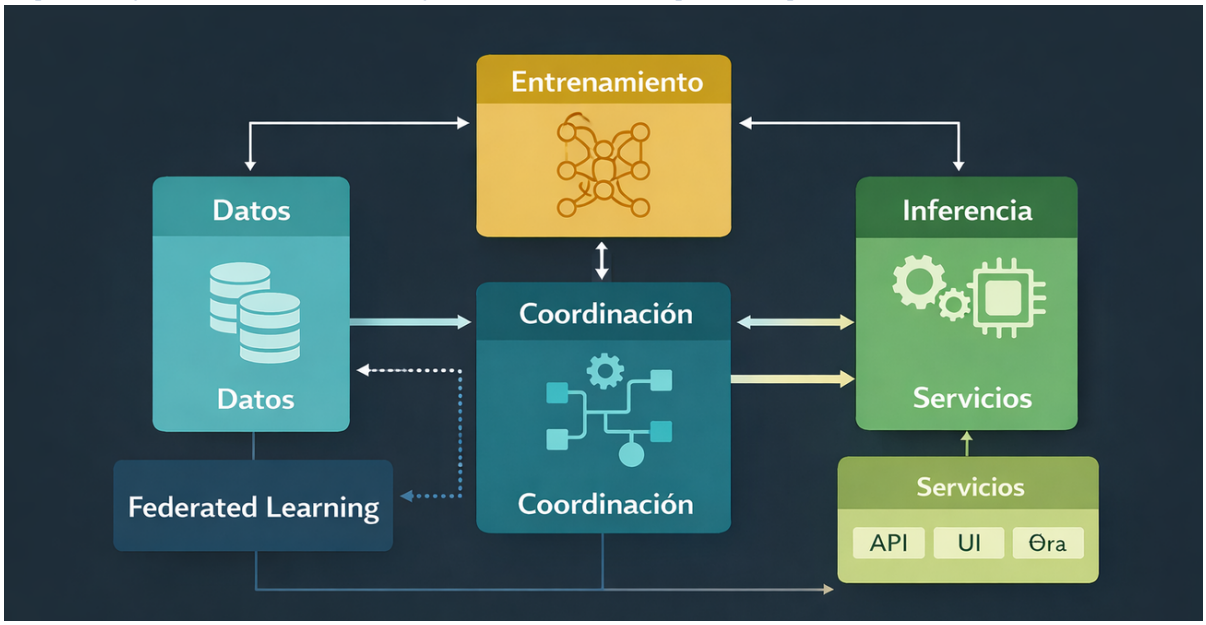
La literatura reciente evidencia que los patrones arquitectónicos continúan desempeñando un papel fundamental en sistemas de IA, ya que influyen directamente en atributos de calidad como el rendimiento, la escalabilidad, la mantenibilidad y la confiabilidad. En particular, estudios centrados en arquitecturas para aprendizaje federado y sistemas distribuidos basados en IA demuestran que la selección de un patrón arquitectónico adecuado puede mejorar significativamente el comportamiento global del sistema, incluso cuando los algoritmos de aprendizaje permanecen inalterados (Compagnucci et al., 2026).

Entre los patrones arquitectónicos más utilizados en sistemas basados en IA se encuentran las arquitecturas cliente-servidor, los estilos basados en microservicios y las arquitecturas orientadas a eventos. En el contexto del aprendizaje federado, por ejemplo, el patrón cliente-servidor permite centralizar la agregación de modelos, mientras que enfoques más distribuidos favorecen la escalabilidad y la tolerancia a fallos, a costa de un mayor consumo de recursos y complejidad de coordinación. Estos *trade-offs* ponen de manifiesto que las decisiones arquitectónicas tienen un impacto directo en métricas técnicas y operativas del sistema.

La ***Figura 4*** ilustra una arquitectura general de un sistema de inteligencia artificial basado en patrones arquitectónicos, destacando la interacción entre componentes de datos, módulos de entrenamiento, servicios de inferencia y mecanismos de coordinación. Este tipo de arquitectura evidencia cómo los patrones estructurales permiten organizar de manera coherente los distintos elementos del sistema, facilitando su evolución y mantenimiento a lo largo del tiempo.

Figura 4

Arquitectura general de un sistema de inteligencia artificial basada en patrones arquitectónico



Fuente: Elaboración propia.

Desde una perspectiva analítica, los estudios revisados destacan que no existe un patrón arquitectónico universalmente óptimo para sistemas de IA. En su lugar, la literatura enfatiza la necesidad de evaluar cuidadosamente los compromisos entre distintos atributos de calidad en función del contexto de aplicación. **La *Tabla 5*** resume la relación entre patrones arquitectónicos comúnmente utilizados en sistemas de IA y su impacto sobre métricas relevantes, evidenciando que la arquitectura actúa como un factor determinante en el desempeño y la escalabilidad del sistema.

Tabla 5

Relación entre patrones arquitectónicos y métricas de calidad en sistemas basados en IA

Patrón arquitectónico	Atributos de calidad y consideraciones	Trade-offs identificados
Cliente-servidor	Control centralizado y consistencia del modelo	Punto único de fallo y limitaciones de escalabilidad
Microservicios	Escalabilidad y despliegue independiente	Mayor complejidad operativa y sobrecarga de comunicación
Arquitectura orientada a eventos	Alta reactividad y desacoplamiento	Complejidad en el monitoreo y depuración del sistema

Fuente: Elaboración propia.

Los patrones arquitectónicos continúan siendo un elemento clave en el diseño de sistemas basados en inteligencia artificial. Los resultados reportados en la literatura confirman que la arquitectura no solo condiciona el comportamiento funcional del sistema, sino que también influye de manera significativa en métricas técnicas y operativas. Esta evidencia refuerza la necesidad de considerar explícitamente los patrones arquitectónicos durante el diseño de soluciones basadas en IA, especialmente en escenarios donde se busca integrar estos sistemas con procesos de generación de código asistida por inteligencia artificial.

Brecha entre Patrones Arquitectónicos y Generación de Código Asistida por Inteligencia Artificial

A pesar de los avances significativos en el uso de inteligencia artificial generativa dentro de la ingeniería de software, la literatura revisada pone de manifiesto la existencia de una brecha relevante entre los enfoques actuales de generación de código asistida por IA y la aplicación explícita de principios y patrones arquitectónicos. Si bien las herramientas basadas en modelos de lenguaje de gran escala han demostrado ser eficaces para generar código funcional y apoyar tareas de desarrollo, su enfoque predominante continúa centrado en el nivel de implementación, dejando en un segundo plano las decisiones arquitectónicas que condicionan la calidad global del sistema.

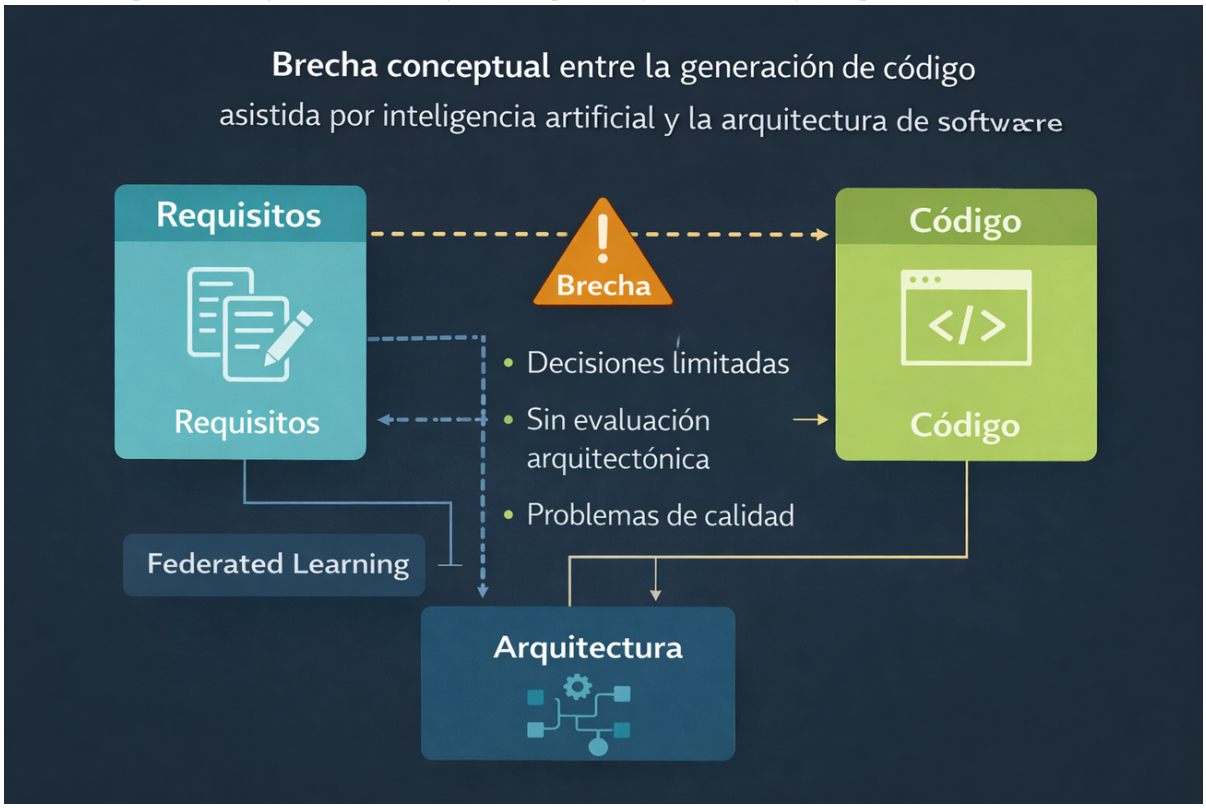
Como se analizó en las secciones previas, la arquitectura de software desempeña un rol central como elemento de enlace entre los requisitos del sistema y su implementación técnica. No obstante, los enfoques

actuales de generación de código asistida por IA rara vez incorporan modelos arquitectónicos explícitos o patrones arquitectónicos como parte de su conocimiento interno. Esta ausencia limita la capacidad de la IA para razonar sobre aspectos estructurales del sistema, tales como la organización de componentes, la distribución de responsabilidades o los *trade-offs* asociados a distintos atributos de calidad.

La ***Figura 5*** ilustra de manera conceptual esta brecha, mostrando cómo las herramientas de IA generativa suelen establecer una conexión directa entre los requisitos y el código, omitiendo o simplificando excesivamente la capa arquitectónica. Esta desconexión puede dar lugar a soluciones que, aunque funcionalmente correctas, presentan problemas de mantenibilidad, escalabilidad o coherencia estructural a largo plazo.

Figura 5

Brecha conceptual entre la generación de código asistida por inteligencia artificial y la arquitectura de software



Fuente: Elaboración propia.

Asimismo, la literatura destaca que la falta de integración sistemática entre requisitos, arquitectura y código dificulta la evaluación y validación de las decisiones generadas por la IA. Sin mecanismos que permitan representar y verificar explícitamente la arquitectura, resulta complejo garantizar que el código producido cumpla con los atributos de calidad esperados o que sea coherente con las restricciones del dominio. Esta situación se agrava en sistemas complejos y distribuidos, donde las decisiones arquitectónicas tienen un impacto acumulativo a lo largo del ciclo de vida del software.

En este contexto, diversos autores coinciden en la necesidad de avanzar hacia enfoques que integren el conocimiento arquitectónico como un elemento explícito dentro de los procesos asistidos por inteligencia artificial. La incorporación de patrones arquitectónicos, modelos estructurales y criterios de evaluación permitiría a los sistemas de IA generativa razonar de manera más informada sobre el diseño del software, reduciendo el riesgo de decisiones subóptimas y mejorando la calidad de las soluciones generadas.

La brecha identificada entre los patrones arquitectónicos y la generación de código asistida por IA representa una oportunidad de investigación relevante. Abordar esta brecha no solo contribuiría a fortalecer la coherencia entre requisitos, arquitectura y código, sino que también permitiría avanzar hacia herramientas de IA generativa más robustas, confiables y alineadas con los principios fundamentales de la ingeniería de software. Esta problemática constituye el punto de partida para el análisis presentado en las secciones de resultados y discusión de este trabajo.

RESULTADOS

En esta sección se presentan los resultados obtenidos a partir del análisis comparativo de investigaciones recientes relacionadas con la integración de inteligencia artificial generativa y arquitectura de software. Al tratarse de una revisión del estado del arte, los resultados corresponden a hallazgos reportados por estudios previos y se organizan en función de distintos enfoques arquitectónicos y niveles de integración de la IA en el proceso de desarrollo de software.

Resultados sobre arquitecturas para sistemas basados en inteligencia artificial

Los estudios analizados coinciden en que los sistemas basados en inteligencia artificial generativa requieren arquitecturas de software explícitas y bien estructuradas para garantizar su correcto funcionamiento. En particular, se ha evidenciado que arquitecturas modulares y orientadas a componentes facilitan la integración de modelos de lenguaje de gran escala, permitiendo separar responsabilidades como la gestión de datos, el control del modelo, la validación de resultados y la interacción con el usuario (Erickson et al., 2025).

Los resultados reportados indican que estas arquitecturas contribuyen significativamente a mejorar la mantenibilidad y la escalabilidad del sistema, especialmente en escenarios donde los modelos deben actualizarse o ajustarse de manera frecuente. Asimismo, se observa que una arquitectura bien definida facilita la incorporación de mecanismos de monitoreo, trazabilidad y control, aspectos críticos en sistemas inteligentes que operan en entornos reales y regulados.

A continuación, se sintetizan los principales tipos de arquitecturas identificadas en la literatura y los atributos de calidad que se ven favorecidos por cada una de ellas:

Tabla 6

Arquitecturas de software utilizadas en sistemas basados en IA generativa y atributos de calidad favorecidos

Tipo de arquitectura	Características principales	Atributos de calidad favorecidos
Arquitectura modular basada en componentes	Separación clara de responsabilidades entre datos, modelos, servicios e interfaz	Mantenibilidad, reutilización, escalabilidad
Arquitectura orientada a servicios / microservicios	Despliegue independiente y comunicación mediante APIs	Escalabilidad, flexibilidad, tolerancia a fallos
Arquitectura con conocimiento estructurado	Integración de grafos de conocimiento y mecanismos de recuperación de información	Coherencia, trazabilidad, reducción de alucinaciones

Fuente: Elaboración propia.

Resultados sobre la integración de IA generativa en decisiones Arquitectónicas

Otro conjunto relevante de resultados se relaciona con el uso de inteligencia artificial generativa como apoyo directo en la toma de decisiones arquitectónicas. La literatura analizada muestra que los modelos generativos pueden asistir en la identificación de estilos arquitectónicos adecuados a partir de requisitos funcionales y no funcionales, así como en la evaluación preliminar de alternativas de diseño (Esposito et al., 2026a).

No obstante, los estudios coinciden en que la efectividad de estos enfoques depende en gran medida de la calidad del contexto proporcionado al modelo y del conocimiento arquitectónico incorporado explícitamente en el sistema. En ausencia de estas condiciones, se ha reportado el riesgo de obtener decisiones arquitectónicas inconsistentes, difíciles de justificar o que no consideran adecuadamente los trade-offs entre atributos de calidad.

Resultados sobre el uso de conocimiento estructurado en arquitecturas con IA

La integración de conocimiento estructurado, como grafos de conocimiento, ontologías o repositorios arquitectónicos, emerge como uno de los resultados más significativos identificados en la literatura. Estudios recientes demuestran que las arquitecturas que combinan modelos de lenguaje con mecanismos de recuperación de información presentan mejoras sustanciales en términos de precisión, coherencia y reducción de alucinaciones (Erickson et al., 2025).

Estos resultados evidencian que la arquitectura actúa como un elemento mediador entre el modelo generativo y las fuentes de conocimiento confiables, permitiendo al sistema razonar de manera más contextualizada. Además, se ha observado que este tipo de arquitecturas favorece la trazabilidad de las decisiones generadas por la IA, aspecto fundamental para entornos críticos y regulados.

Resultados comparativos y limitaciones identificadas

A partir del análisis comparativo de los estudios revisados, se identifican una serie de beneficios y limitaciones comunes en las soluciones actuales. A continuación, se resumen los principales enfoques de integración de IA generativa en arquitectura de software, junto con sus ventajas y limitaciones más relevantes:

Tabla 7

Comparación de enfoques de integración de IA generativa en arquitectura de software

Enfoque	Principales beneficios	Limitaciones identificadas
Generación de código asistida por IA	Aumento de productividad y automatización de tareas repetitivas	Desconexión con decisiones arquitectónicas
IA para decisiones arquitectónicas	Apoyo en fases tempranas de diseño y análisis de alternativas	Dependencia del contexto y falta de explicabilidad
IA con conocimiento arquitectónico explícito	Mayor coherencia, trazabilidad y control de calidad	Necesidad de repositorios arquitectónicos y validación especializada

Fuente: Elaboración propia.

En función del análisis comparativo realizado, se identifica la necesidad de integrar de manera explícita los requisitos, las decisiones arquitectónicas y la generación de código asistida por inteligencia artificial. Los resultados muestran que los enfoques que incorporan arquitectura de software como un elemento central presentan ventajas claras frente a soluciones centradas exclusivamente en la generación automática de código.

A pesar de las limitaciones identificadas, los resultados analizados coinciden en que la incorporación explícita de arquitectura de software en sistemas de inteligencia artificial generativa representa un avance significativo frente a enfoques puramente orientados a la implementación. Estos hallazgos sientan las bases para el análisis crítico desarrollado en la siguiente sección de discusión.

DISCUSIÓN Y TRABAJOS FUTUROS

La revisión del estado del arte presentada en este trabajo pone de manifiesto que la inteligencia artificial generativa ha comenzado a desempeñar un rol relevante en distintas fases del ciclo de vida del software, particularmente en la ingeniería de requisitos, el diseño preliminar y la generación automática de código. Sin embargo, el análisis crítico de la literatura evidencia que la integración explícita de patrones arquitectónicos dentro de estos procesos sigue siendo limitada y, en muchos casos, implícita o informal (Esposito et al., 2026a).

Los estudios revisados muestran que los modelos de lenguaje de gran escala son capaces de asistir eficazmente en tareas de bajo y medio nivel, como la generación de fragmentos de código, la documentación técnica o la identificación de requisitos funcionales. Investigaciones recientes confirman que los LLMs pueden mejorar la productividad del desarrollo de software cuando se emplean como herramientas de apoyo dentro de procesos controlados y supervisados (Bhalla & Jodhka, 2025). No obstante, cuando estas herramientas se utilizan sin una guía arquitectónica clara, existe un riesgo significativo de producir soluciones inconsistentes, difíciles de mantener o que no cumplen adecuadamente con los atributos de calidad esperados (Al-Azzoni et al., 2026), (Esposito et al., 2026a). Esta situación refuerza la idea de que la arquitectura de software continúa siendo un elemento central que no puede ser completamente automatizado sin comprometer la calidad global del sistema.

Desde una perspectiva arquitectónica, los resultados analizados confirman que los patrones arquitectónicos siguen siendo fundamentales incluso en sistemas basados en inteligencia artificial. Estudios recientes demuestran que la elección de un patrón arquitectónico adecuado tiene un impacto directo en métricas clave como el rendimiento, la escalabilidad, la mantenibilidad y la seguridad del sistema (Esposito et al., 2026a) (Shaon & Akter, 2025). En particular, investigaciones orientadas a la detección automática de vulnerabilidades evidencian que la ausencia de una estructura arquitectónica explícita puede amplificar riesgos de seguridad y dificultar la validación de soluciones generadas automáticamente (Shaon & Akter, 2025). En este sentido, la arquitectura actúa como un mecanismo de control que permite gestionar la complejidad inherente a los sistemas inteligentes y mitigar los riesgos asociados a la automatización excesiva.

Uno de los principales desafíos identificados en la literatura es la ausencia de mecanismos formales que permitan a los sistemas de IA generativa razonar de manera explícita sobre decisiones arquitectónicas. La

mayoría de las soluciones actuales carecen de modelos arquitectónicos explícitos o de representaciones estructuradas del conocimiento arquitectónico, lo que limita la capacidad de la IA para comprender el impacto de sus decisiones más allá del nivel de implementación (Al-Azzoni et al., 2026) , (Esposito et al., 2026a). Revisiones recientes señalan además problemas relacionados con la explicabilidad, la trazabilidad y la generalización de los modelos, especialmente en sistemas complejos y multimodales (Ajayi et al., 2025). Esta brecha conceptual constituye una de las principales oportunidades de investigación futura.

En cuanto a los trabajos futuros, se identifican varias líneas de investigación prometedoras. En primer lugar, resulta necesario desarrollar enfoques que integren patrones arquitectónicos como conocimiento explícito dentro de los modelos de IA generativa, ya sea mediante ontologías, grafos de conocimiento o repositorios arquitectónicos estructurados que permitan razonar sobre decisiones de alto nivel (Esposito et al., 2026a). En segundo lugar, se requiere avanzar en la definición de métricas y métodos de evaluación que permitan medir de manera objetiva la calidad arquitectónica del código generado automáticamente, complementando métricas tradicionales de corrección funcional con indicadores de mantenibilidad, seguridad y robustez estructural (Shaon & Akter, 2025).

Finalmente, futuras investigaciones deberían explorar marcos híbridos en los que la inteligencia artificial actúe como un asistente del arquitecto de software, manteniendo la intervención humana en decisiones críticas y aprovechando la automatización para tareas repetitivas o de soporte. Estudios recientes sugieren que este enfoque colaborativo puede mejorar la confianza en los sistemas basados en IA y reducir los riesgos asociados a la generación automática sin supervisión adecuada (Bhalla & Jodhka, 2025), (Ajayi et al., 2025).

La convergencia entre arquitectura de software e inteligencia artificial generativa representa un campo emergente con un alto potencial de impacto. No obstante, para que esta convergencia sea efectiva y sostenible, es imprescindible avanzar hacia enfoques que reconozcan explícitamente la importancia de los patrones arquitectónicos y los integren de manera sistemática en los procesos de generación de código asistida por inteligencia artificial.

CONCLUSIONES

La revisión del estado del arte realizada en este estudio permite concluir, en primer lugar, que la arquitectura de software continúa siendo un pilar fundamental en el desarrollo de sistemas complejos, incluso en escenarios donde se incorpora inteligencia artificial generativa. Los hallazgos analizados confirman que las decisiones arquitectónicas influyen de manera directa en atributos de calidad críticos como la mantenibilidad, la escalabilidad y la robustez, los cuales no pueden garantizarse únicamente mediante la generación automática de código.

Se concluye que los patrones arquitectónicos representan un mecanismo clave para capturar y reutilizar conocimiento experto, permitiendo estructurar los sistemas de forma coherente y reducir los riesgos asociados a soluciones improvisadas o excesivamente centradas en la implementación. Este resultado amplía los hallazgos de estudios previos al evidenciar que, incluso en contextos altamente automatizados, los patrones arquitectónicos siguen siendo determinantes para la calidad global del software.

Los resultados de la literatura revisada muestran que la inteligencia artificial generativa posee un alto potencial para apoyar diversas actividades de la ingeniería de software, particularmente en la ingeniería de requisitos, el diseño preliminar y la generación de código. Sin embargo, se observa que su aplicación actual se concentra mayoritariamente en niveles de implementación, dejando en un segundo plano la incorporación explícita de decisiones arquitectónicas, lo que limita el impacto positivo de estas tecnologías en sistemas de mayor complejidad.

Otro hallazgo relevante es que la integración de IA generativa en procesos arquitectónicos puede acelerar el diseño y servir como apoyo para arquitectos con menor experiencia, siempre que se disponga de una arquitectura de software bien definida y de conocimiento estructurado que guíe el comportamiento del modelo. En este sentido, los resultados analizados sugieren que la IA generativa resulta más efectiva cuando actúa como un asistente del proceso arquitectónico y no como un sustituto del razonamiento humano.

Por otra parte, la revisión evidencia una brecha significativa en la investigación actual, asociada a la ausencia de representaciones arquitectónicas explícitas y de mecanismos formales para evaluar las decisiones arquitectónicas generadas por IA. Esta limitación afecta la confiabilidad, la explicabilidad y la validación de las soluciones propuestas, especialmente en sistemas complejos y distribuidos, ampliando y profundizando

observaciones ya planteadas en estudios anteriores.

Este trabajo concluye que es necesario avanzar hacia enfoques híbridos que integren patrones arquitectónicos como conocimiento explícito dentro de los sistemas de inteligencia artificial generativa, manteniendo la intervención humana en decisiones críticas. Dado que los resultados obtenidos corresponden a una revisión del estado del arte, estas conclusiones deben considerarse preliminares, lo que sugiere la necesidad de futuros estudios orientados al desarrollo de modelos, métricas y herramientas que permitan evaluar de manera objetiva la calidad arquitectónica del código generado por IA y su impacto en el ciclo de vida del software.

FINANCIACIÓN

La presente investigación no contó con financiamiento externo por parte de organizaciones gubernamentales, universidades, centros de investigación, becas ni proyectos institucionales. El estudio fue desarrollado de manera independiente por los autores, sin apoyo económico específico.

CONFLICTO DE INTERESES

Los autores declaran que no existe ningún conflicto de intereses de carácter personal, académico, financiero o institucional que pudiera haber influido en el desarrollo o en los resultados de la presente investigación.

REFERENCIAS

Abrahão, S., Grundy, J., Pezzè, M., Storey, M.-A., & Tamburri, D. A. (2025). Software Engineering by and for Humans in an AI Era. En ACM Transactions on Software Engineering and Methodology (Vol. 34, Número 5). https://doi.org/10.1145/3715111

Ajayi, O. O., Kurien, A. M., Djouani, K., & Dieng, L. (2025). A Multimodal Systematic Review of Drivers' Fatigue Detection Methodologies, Datasets, and Models. En IEEE Access (Vol. 13, pp. 158266-158284). https://doi.org/10.1109/ACCESS.2025.3606900

Al-Azzoni, I., Iqbal, S., Al Ashkar, T., & Erum, Z. (2026). Integrating Model-Driven Engineering and Large Language Models for Test Scenario Generation for Smart Contracts. En Information (Switzerland) (Vol. 17, Número 1). https://doi.org/10.3390/info17010001

Autili, M., De Sanctis, M., Inverardi, P., Memon, M. A., Pelliccione, P., & Pettinari, S. (2026). A reference architecture for ethical-aware autonomous systems. En Journal of Systems and Software (Vol. 235). https://doi.org/10.1016/j.jss.2025.112749

Bhalla, J. S., & Jodhka, M. K. (2025). Leveraging Large Language Models in the Software Development Lifecycle: Opportunities and Challenges. En International Journal of Advanced Computer Science and Applications (Vol. 16, Número 11, pp. 43-51). https://doi.org/10.14569/IJACSA.2025.0161105

Cheng, H., Husen, J. H., Lu, Y., Racharak, T., Yoshioka, N., Ubayashi, N., & Washizaki, H. (2026). Generative AI for Requirements Engineering: A Systematic Literature Review. En Software—Practice and Experience (Vol. 56, Número 2, pp. 141-170). https://doi.org/10.1002/spe.70029

Compagnucci, I., Pinciroli, R., & Trubiani, C. (2026). Experimenting Architectural Patterns in Federated Learning Systems. En Journal of Systems and Software (Vol. 232). https://doi.org/10.1016/j.jss.2025.112655

Dhawan, M. (2026). Decoupling Deployment Velocity From Platform Governance: An Empirical Case Study of WebSocket-Enabled Gateway-Microservice Architecture. En IEEE Access (Vol. 14, pp. 9348-9358). https://doi.org/10.1109/ACCESS.2026.3654150

Erickson, J. S., Santos, H., Pinheiro, V., McCusker, J. P., & McGuinness, D. L. (2025). LLM experimentation through knowledge graphs: Towards improved management, repeatability, and verification. En Journal of Web Semantics (Vol. 85). https://doi.org/10.1016/j.websem.2024.100853

Esposito, M., Li, X., Moreschini, S., Ahmad, N., Cerny, T., Vaidhyanathan, K., Lenarduzzi, V., & Taibi, D. (2026a). Generative AI for software architecture. Applications, challenges, and future directions. En Journal of Systems and Software (Vol. 231). https://doi.org/10.1016/j.jss.2025.112607

Esposito, M., Li, X., Moreschini, S., Ahmad, N., Cerny, T., Vaidhyanathan, K., Lenarduzzi, V., & Taibi, D. (2026b). Generative AI for software architecture. Applications, challenges, and future directions. En Journal of Systems and Software (Vol. 231). https://doi.org/10.1016/j.jss.2025.112607

Fuentes-Quijada, G., Ruiz-González, F., & Caro, A. (2025). Enterprise Architecture and IT Governance to Support the BizDevOps Approach: A Systematic Mapping Study. En Information Systems Frontiers (Vol. 27, Número 3, pp. 865-888). https://doi.org/10.1007/s10796-024-10473-2

Gacitúa, R., Pereira, J., & Klafft, M. (2026). Explainability in Software Architectural Decisions: The ADR-E Framework and Empirical Evaluation. En IEEE Access (Vol. 14, pp. 9038-9061). https://doi.org/10.1109/ACCESS.2025.3648573

Hyun, S., & Hurtado, J. A. (2023). Traceability of Architectural Design Decisions and Software Artifacts: A Systematic Mapping Study. En Foundations of Computing and Decision Sciences (Vol. 48, Número 4, pp. 401-423). https://doi.org/10.2478/fcds

-2023-0018

Le, D. M., Dang, D.-H., & Vo, H. D. (2025). Layered microservices architecture: A multitree-based domain-driven approach. En Information and Software Technology (Vol. 183). https://doi.org/10.1016/j.infsof.2025.107720

Liu, Z., Cheon, S., Stanbury, A., Jiao, X., Xing, W., & Kang, H. (2026). Towards contextual-based AI: A scoping review of artificial intelligence in X reality for personalized learning. En Computers and Education: Artificial Intelligence (Vol. 10). https://doi.org/10.1016/j.caeai.2025.100523

Nguyen, V.-V., Nguyen, H.-K., Nguyen, K.-S., Luong, T. M.-H., Vu, D.-Q., Phung, T.-N., & Nguyen, T.-V. (2026). A Novel Unified Framework for Automated Generation and Multimodal Validation of UML Diagrams. En CMES - Computer Modeling in Engineering and Sciences (Vol. 146, Número 1). https://doi.org/10.32604/cmes.2025.075442

Nouman, M., Azam, M., Saleh, A. M., Alsaeedi, A., & Abuaddous, H. Y. (2023). A systematic review of non-functional requirements mapping into architectural styles. En Bulletin of Electrical Engineering and Informatics (Vol. 12, Número 2, pp. 1226-1236). https://doi.org/10.11591/eei.v12i2.4081

Oruthotaarachchi, C., & Wijayanayake, J. (2025). Aligning Software Product Management with Software Engineering Concepts: A Systematic Literature Review. En Journal of Information Systems Engineering and Business Intelligence (Vol. 11, Número 2, pp. 143-159). https://doi.org/10.20473/jisebi.11.2.143-159

Rodrigues, H., Rito Silva, A., & Avritzer, A. (2025). Assessment of performance and its scalability in microservice architectures: Systematic literature review. En Journal of Systems and Software (Vol. 230). https://doi.org/10.1016/j.jss.2025.112500

Shaon, Md. S. H., & Akter, M. S. (2025). Modern Approaches to Software Vulnerability Detection: A Survey of Machine Learning, Deep Learning, and Large Language Models. En Electronics (Switzerland) (Vol. 14, Número 22). https://doi.org/10.3390/electronics14224449