Alianza de Investigadores Internacionales S.A.S ALININ

Fondo Editorial Alianza de Investigadores Internacionales ALININ



Revista Temario Científico

Vol. 6 Nro. 2. Julio - Diciembre, 2026
ISSN 2805-850X

Esta obra está bajo una Licencia Creative Commons Atribución-No Comercial-Compartir Igual 4.0 Internacional

Revista Temario Científico

Publicación continua, Julio-Diciembre, 2026
E-ISSN 2805-850X
Alianza de Investigadores Internacionales, SAS (ALININ)

Cómo citar:
Espejo, Y. (2026). Modernización de sistemas legacy con inteligencia artificial generativa: evaluación de la transformación de especificaciones funcionales en servicios. *Revista Temario Científico*, *6*(2). e20260602. https://doi.org/10.47212/rtcAlinin.4.226.10

MODERNIZACIÓN DE SISTEMAS LEGACY CON INTELIGENCIA ARTIFICIAL GENERATIVA: EVALUACIÓN DE LA TRANSFORMACIÓN DE ESPECIFICACIONES FUNCIONALES EN SERVICIOS

Yeismer Cornelio Espejo Bohórquez

Espejo, Y.

Especialista en Ingeniería de Software. Universidad de La Salle. Bogotá, D. C., Colombia

Resumen

Este estudio evalúa un enfoque basado en inteligencia artificial generativa para transformar especificaciones funcionales de un sistema legacy en servicios de software, y determinar su cumplimiento funcional, utilidad arquitectónica, incidencias y productividad estimada frente al desarrollo tradicional. Se realizó un estudio de caso empírico observacional sobre un sistema legacy real, cuya población fue de 1.280 servicios generados con el modelo Claude Opus 4.6 a partir de especificaciones estructuradas. Mediante muestreo intencional por complejidad se seleccionaron los 500 servicios con tres o más Story Points. Los datos se recolectaron con pruebas automatizadas, revisión manual, la herramienta de incidencias del proyecto y una rúbrica de cinco criterios para valorar la utilidad, aplicada por el arquitecto líder y cerrada por consenso de tres arquitectos, sin cálculo de concordancia entre evaluadores. Los tiempos se estimaron a partir de la velocidad histórica del equipo. De los 500 servicios, 425 (85 %) cumplieron las especificaciones sin ajustes y 75 requirieron iteración; se identificaron 85 incidencias, con predominio de las arquitectónicas (41 %), y 430 servicios (86 %) se clasificaron como útiles. El tiempo estimado por componente fue de 4 días con el enfoque generativo y de 10 días con el tradicional. Se concluye que la inteligencia artificial generativa puede acelerar la transformación de sistemas legacy y producir servicios que cumplen las especificaciones y resultan útiles, siempre que se integre con validación técnica, evaluación arquitectónica, iteración y supervisión humana especializada. Al tratarse de un único sistema y de tiempos estimados, los resultados deben interpretarse con cautela.

Palabras clave

Especificaciones funcionales, inteligencia artificial generativa, modernización, servicios, sistemas legacy

MODERNIZATION OF LEGACY SYSTEMS WITH GENERATIVE ARTIFICIAL INTELLIGENCE: EVALUATION OF THE TRANSFORMATION OF FUNCTIONAL SPECIFICATIONS INTO SERVICES

Abstract

This study evaluates a generative artificial intelligence approach for transforming the functional specifications of a legacy system into software services, and determines their functional compliance, architectural usefulness, incidents, and estimated productivity compared with traditional development. An observational empirical case study was conducted on a real legacy system, whose population consisted of 1,280 services generated with the Claude Opus 4.6 model from structured specifications. Through purposive sampling by complexity, the 500 services with three or more Story Points were selected. Data were collected through automated tests, manual review, the project's incident tool, and a five-criterion rubric to assess usefulness, applied by the lead architect and settled by consensus among three architects, without computing inter-rater agreement. Times were estimated from the team's historical velocity. Of the 500 services, 425 (85%) met the specifications without adjustments and 75 required iteration; 85 incidents were identified, predominantly architectural (41%), and 430 services (86%) were classified as useful. The estimated time per component was 4 days with the generative approach and 10 days with the traditional one. It is concluded that generative artificial intelligence can accelerate the transformation of legacy systems and produce services that meet the specifications and are useful, provided it is integrated with technical validation, architectural assessment, iteration, and specialized human supervision. Because the study covers a single system and relies on estimated times, the results should be interpreted with caution.

Keywords

Functional specifications, generative artificial intelligence, legacy systems, modernization, services

https://doi.org/10.47212/rtcAlinin.4.226.10

Recibido: : 20-07-2026
Aceptado: 21-10-2026
Publicado: 31-10-2026

CC BY NC SA

Introducción

Los sistemas *legacy* continúan siendo relevantes para numerosas organizaciones debido al conocimiento funcional y las reglas de negocio acumuladas durante su evolución (Irani *et al.*, 2023). Sin embargo, presentan problemas recurrentes: alto acoplamiento entre componentes, costos elevados de mantenimiento (Crotty & Horrocks, 2017), baja flexibilidad ante nuevos requisitos y dependencia de conocimiento tácito concentrado en pocos expertos (Hasan *et al.*, 2023; Sandborn 6 Prabhakar, 2015).

Además, la migración es un proceso difícil y costoso, y la falta de comprensión y preparación figura entre las causas de muchos fracasos (Gholami *et al.*, 2017), lo que dificulta estimar con confianza el esfuerzo necesario para transformar el sistema.

Para afrontar estos problemas, la literatura reporta técnicas de modernización amplias, pero poco consolidadas, que van desde el envolvimiento de componentes hasta la reescritura completa y la descomposición hacia microservicios (Fávero *et al.*, 2025). Estudios previos señalan que rara vez se documentan como un proceso completo y reproducible (Bakar *et al.*, 2019; Pérez *et al.*, 2011), y que persiste la escasez de evidencia empírica sobre los beneficios reales de cada enfoque (Fávero *et al.*, 2025).

La aparición de los modelos de lenguaje de gran escala (LLM, por sus siglas en inglés) ha modificado el panorama de la ingeniería de software. Una revisión sistemática de 395 artículos publicados entre 2017 y 2024 evidencia su adopción creciente en tareas antes manuales, como la generación de código, de pruebas y de documentación (Hou *et al.*, 2024), lo que abre la posibilidad de automatizar parcialmente procesos de transformación de software (Nguyen-Duc *et al.*, 2023).

La evidencia sobre generación de código se ha consolidado sobre todo con *benchmarks* como HumanEval (Chen *et al.*, 2021), aunque sus resultados no siempre se trasladan a código *legacy* o de dominio específico. En la migración de código *legacy* hacia lenguajes modernos, estudios recientes muestran que los LLM pueden capturar la lógica de negocio original y producir implementaciones funcionalmente equivalentes en un porcentaje significativo de los casos evaluados (Gandhi *et al.*, 2024).

No obstante, la mayor parte de esa evidencia proviene de lenguajes y tareas modernas, y los estudios sobre sistemas *legacy*, en particular sobre procesos de transformación completos y no solo traducción de fragmentos, siguen siendo escasos (Diggs *et al.*, 2024).

En la literatura consultada se encontraron pocos estudios que evalúen un volumen considerable de especificaciones funcionales de un sistema *legacy* real y reporten, de forma conjunta, el cumplimiento funcional, los tipos de incidencias técnicas y la productividad frente a un proceso tradicional.

Comprender en qué medida y en qué condiciones esta tecnología puede apoyar la modernización es relevante para las organizaciones que deben decidir si adoptarla. Este estudio busca aportar evidencia empírica a ese vacío. Entendiendo que la Inteligencia Artificial (IA) se ha desarrollado desde distintas perspectivas y modelos, siendo los más prometedores son aquellos que intentan imitar al ser humano o aquellos que generan producción a partir del ideal racional (Duarte y Chacon, 2022), el objetivo de la investigación es evaluar un enfoque basado en inteligencia artificial generativa para transformar especificaciones funcionales de un sistema *legacy* real en servicios de software, mediante un estudio de caso empírico observacional de 500 servicios con tres o más *Story Points*. Se analizan el cumplimiento funcional, la validación técnica y arquitectónica, la clasificación de incidencias y la productividad estimada frente a un proceso de desarrollo tradicional.

Metodología

Diseño de la investigación

El estudio se desarrolló como un estudio de caso empírico observacional (Runeson y Höst, 2009), con un componente de evaluación estructurada, sobre un proyecto real de transformación de un sistema *legacy*. No corresponde a un diseño experimental, porque no hubo asignación aleatoria de las especificaciones a los enfoques comparados ni manipulación controlada de variables. Su propósito fue evaluar el comportamiento de un enfoque basado en inteligencia artificial generativa al transformar especificaciones funcionales existentes en servicios de software, en dos dimensiones: la calidad de los servicios generados (cumplimiento funcional, validación técnica y arquitectónica, e incidencias) y la productividad, estimada como el tiempo de desarrollo frente a un proceso tradicional de referencia.

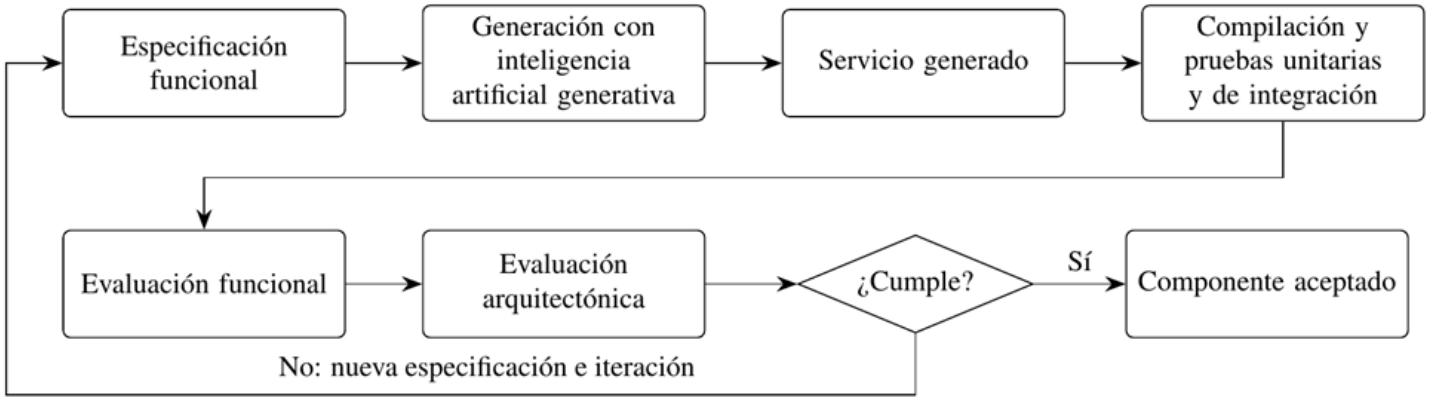
Ambas dimensiones, evaluadas de forma conjunta, constituyen el marco de evaluación multidimensional del estudio, que integra cinco componentes: el cumplimiento funcional inicial (IFCR), la validación técnica (compilación y pruebas unitarias y de integración), la utilidad arquitectónica (rúbrica de cinco criterios), la clasificación de incidencias por tipo y el tiempo de desarrollo estimado. La unidad de análisis fue el servicio o componente generado a partir de cada especificación. La secuencia general (Figura 1) comprendió la generación, la evaluación funcional, la validación técnica, la evaluación arquitectónica y la iteración de los componentes que no cumplieron. De este modo, la inteligencia artificial generativa se empleó dentro de un proceso supervisado y no como un mecanismo autónomo.

Población y muestra

La población estuvo constituida por los 1.280 servicios generados a partir de las 1.280 especificaciones funcionales del sistema legacy objeto de estudio; cada especificación produjo exactamente un servicio. La complejidad de cada

Figura 1

Proceso de transformación y evaluación de los servicios



Fuente: elaboración propia (2026).

servicio se midió con los Story Points asignados por el equipo del proyecto antes de la generación (Tawosi et al., 2023), en una escala de Fibonacci (1, 2, 3, 5, 8), por lo que reflejan la complejidad estimada a priori y no el esfuerzo posterior.

La muestra fue de 500 servicios, seleccionada mediante un muestreo intencional por criterio de complejidad (Etikan et al., 2016): se incluyeron todos los servicios con tres o más Story Points, mientras que los 780 restantes tenían uno o dos (Tabla 1). La muestra no es aleatoria, y las conclusiones se refieren principalmente a servicios de complejidad relativamente mayor.

Configuración técnica del enfoque

La transformación se implementó con Claude Opus 4.6 (Anthropic), accedido mediante la API de Claude a través de la herramienta de orquestación Kilo Code, que coordinó la interacción entre las especificaciones, el código *legacy* y el modelo. En lugar de instrucciones escritas manualmente, un

Tabla 1

Estadísticas descriptivas de los Story Points en los servicios generados

Estadístico	**Total (*n* = 1.280)**	**Seleccionados (*n* = 500)**	**No seleccionados (*n* = 780)**
Mínimo	1	3	1
Máximo	8	8	2
Media	3,04	5,48	1,48
Mediana	2	5	1
Desviación estándar	2,37	2,07	0,50
Percentil 25	1	3	1
Percentil 75	5	8	2

Fuente: elaboración propia (2026)

agente de estandarización transformó el *backlog* del proyecto en especificaciones funcionales detalladas, conformes con el marco OpenSpec, que fueron la entrada principal del proceso generativo.

El código fuente *legacy* se proporcionó como referencia cruzada, para contrastar las especificaciones y los servicios con el comportamiento real del sistema, y se incorporó un mecanismo de generación aumentada por recuperación (RAG, por sus siglas en inglés) (Lewis *et al.*, 2020) basado en *embeddings*, que recuperaba información de la documentación producida durante el desarrollo. El flujo de OpenSpec se ejecutó en seis etapas secuenciales, cada una con una temperatura configurada (Explore, 0,4; Proposal, 0,2; Specs, 0,1; Design, 0,2; Tasks, 0,1; Apply/ Code, 0,2): valores bajos donde se requería fidelidad a la especificación y valores algo mayores en exploración y diseño. Los servicios se implementaron en Java con el *framework* WebFlux, en una arquitectura reactiva orientada a eventos (Margara & Salvaneschi, 2018).

Técnicas e instrumentos de recolección de datos

La recolección de datos se basó en la observación estructurada del proceso y en el análisis de los registros del proyecto. Los mismos 500 componentes se sometieron a cuatro evaluaciones. Primero, el cumplimiento funcional: se verificó, mediante pruebas automatizadas y revisión manual, la correspondencia entre el comportamiento implementado y la especificación de origen, con una clasificación binaria. Un componente se consideró correcto si cumplía la especificación sin requerir cambios; en caso contrario, requirió ajustes y una nueva iteración de la especificación.

Segundo, la validación técnica, con la compilación y las pruebas unitarias y de integración como instrumentos, que aportaron evidencia complementaria para detectar problemas, pero no fueron por sí solas el criterio de cumplimiento.

Tercero, la clasificación de las incidencias detectadas, registradas con la herramienta del proyecto, en sintácticas (estructura o sintaxis del código), semánticas (comportamiento distinto del esperado) y arquitectónicas (inadecuación a los criterios de la arquitectura objetivo). Una incidencia no equivale a un componente: un componente que requirió iteración pudo presentar una o más incidencias.

Cuarto, la evaluación arquitectónica, en la que el arquitecto líder clasificó cada componente como útil o innecesario mediante una rúbrica de cinco criterios binarios: pertinencia funcional (corresponde a una funcionalidad requerida), integración arquitectónica (se ajusta a los patrones y capas definidos), reutilización (evita duplicar funcionalidad existente), cumplimiento de estándares (adhiere a las convenciones de codificación y diseño del proyecto) y necesidad en la arquitectura objetivo (no es redundante ni prescindible). Cada componente obtuvo un puntaje de 0 a 5. El puntaje orientó la decisión, pero la

clasificación final ponderó los criterios incumplidos. Finalmente, los registros históricos de la herramienta de gestión del proyecto (Jira), con los *Story Points* y los días de las historias ya desarrolladas, se emplearon para calcular la *velocity* del equipo.

Procesamiento y análisis de los datos

Se emplearon dos métricas sobre los 500 componentes. La Tasa de Cumplimiento Funcional Inicial (IFCR, por *Initial Functional Compliance Rate*) es la proporción de componentes que cumplieron la especificación sin requerir ajustes:

$$IFCR = \frac{N_{correctos}}{N_{evaluados}} \times 100$$

$$IFCR = \frac{425}{500} \times 100 = 85\,\%$$

La utilidad es la proporción de componentes clasificados como útiles por el equipo de arquitectura:

$$Utilidad = \frac{N_{útiles}}{N_{evaluados}} \times 100$$

$$Utilidad = \frac{430}{500} \times 100 = 86\,\%$$

Ambas miden dimensiones distintas: la IFCR mide el cumplimiento inicial de la especificación, no la tasa final tras las iteraciones, y la utilidad mide la valoración arquitectónica.

Modelo de estimación de tiempos

El tiempo de desarrollo se analizó con un modelo de dos parámetros, calibrado con los registros de la herramienta de gestión del proyecto (Jira) y aplicado a los *Story Points* de cada servicio, y no con la duración individual de cada componente. El primero es la *velocity* histórica del equipo con el enfoque tradicional (v), expresada en días por *Story Point* y calculada con las 500 historias (cada historia corresponde a un servicio) desarrolladas con el enfoque tradicional entre el 1 de febrero y el 1 de mayo de 2026, antes de incorporar la inteligencia artificial generativa, y con días hábiles que corresponden a la duración de cada historia (tiempo transcurrido) y no al esfuerzo-persona.

El segundo, es un factor de aceleración (f), obtenido de los mismos 500 servicios, desarrollados posteriormente con el enfoque generativo, como la razón entre los tiempos medios de ambos enfoques (10/4 = 2,5). Para un servicio i con SP_i *Story Points*, los tiempos con el enfoque tradicional y con el generativo se estimaron como:

$$T_{trad,i} = SP_i \times v$$

$$T_{gen,i} = \frac{T_{trad,i}}{f}$$

El tiempo así estimado corresponde al ciclo completo de trabajo (elaboración de las especificaciones con OpenSpec, generación, validación, corrección, iteración y pruebas). Como cada historia corresponde a un servicio, la comparación es pareada: los mismos 500 servicios se desarrollaron primero con el enfoque tradicional y después con el generativo.

Para comparar ambos enfoques se calcularon la diferencia de tiempos, $\Delta T = T_{tradicional} - T_{generativo}$, y la razón de productividad temporal, $RT = T_{tradicional}$ /

$T_{generativo}$. Por construcción, RT es igual a *f* en todos los servicios; por tanto, la razón de productividad no constituye un resultado independiente del modelo, sino la expresión del factor de aceleración adoptado, y la comparación es una estimación condicionada a ese parámetro y no una contrastación estadística independiente de los tiempos por componente.

Los tiempos medios estimados por componente fueron de 10 días con el enfoque tradicional y de 4 días con el generativo, con una media de 5,48 *Story Points* en los 500 servicios (Tabla 1). Con estos valores, la *velocity* equivalente es de 1,82 días por *Story Point* (10/5,48) con el enfoque tradicional y de 0,73 (4/5,48) con el generativo, y el factor de aceleración es de 2,5 (10/4).

La Tabla 2 desagrega los tiempos según la categoría de *Story Points*, con $v = 1{,}82$ y $f = 2{,}5$. Como el modelo es lineal, la diferencia absoluta de tiempo crece con la complejidad del servicio, mientras que la razón entre ambos enfoques se mantiene constante.

Para examinar la sensibilidad de la estimación al factor adoptado se calcularon escenarios: con factores de 1,5, 2,0 y 3,0, el tiempo generativo medio sería de 6,7, 5,0 y 3,3 días, lo que equivale a reducciones de 33 %, 50 % y 67 %, respectivamente, frente a los 10 días del enfoque tradicional. Como las diferencias entre componentes derivan de esta conversión y no de observaciones independientes, no se realizó análisis inferencial (pruebas de hipótesis, intervalos de confianza o tamaño del efecto). Tampoco se reporta la dispersión de la velocity entre sprints ni la del factor entre historias, por lo que 1,82 días por Story Point y 2,5 deben leerse como valores promedio.

Validez y confiabilidad

Para reducir la subjetividad de la evaluación arquitectónica se empleó una rúbrica estructurada de criterios binarios, y las clasificaciones fueron revisadas y cerradas por consenso por un equipo de tres arquitectos. Al no ser una evaluación ciega e independiente, no se calculó un coeficiente de concordancia entre evaluadores, lo que se reconoce como limitación.

La compilación y las pruebas unitarias y de integración aportaron evidencia técnica verificable sobre los componentes. Por último, el estudio se realizó sobre un único sistema, con tiempos estimados y no medidos de forma independiente,

Tabla 2

Tiempo estimado por servicio según Story Points

Story Points	Servicios (n)	Tradicional (días)	Generativo (días)	ΔT (días)
3	153	5,5	2,2	3,3
5	165	9,1	3,6	5,5
8	182	14,6	5,8	8,8
Total / media (5,48)	500	10,0	4,0	6,0

Fuente: elaboración propia (2026)

por lo que los resultados deben interpretarse en el contexto del proyecto estudiado.

Resultados y discusión

De los 500 servicios evaluados, 425 (85 %) cumplieron la especificación sin ajustes y 75 requirieron iteración; se identificaron 85 incidencias y 430 servicios (86 %) fueron clasificados como útiles. El tiempo estimado por componente fue de 4 días con el enfoque generativo y de 10 días con el tradicional. En los apartados siguientes se presentan los resultados por dimensión y se discuten a la luz de la literatura.

Cumplimiento funcional e iteraciones

De los 500 servicios evaluados, 425 (85 %) cumplieron la especificación funcional sin requerir ajustes, lo que corresponde a una IFCR del 85 %. Los 75 servicios restantes (15 %) requirieron entre 2 y 3 intentos de generación en total, incluido el inicial, con un promedio de 2,51, y todos alcanzaron finalmente el cumplimiento funcional esperado, de modo que ningún componente de la muestra quedó sin resolver. Este resultado debe interpretarse con cautela frente a los *benchmarks* genéricos de generación de código. HumanEval, ampliamente usado para evaluar modelos en tareas de programación desde cero, reportó tasas de éxito mucho menores en los primeros modelos evaluados, como el 28,8 % de Codex en su versión original (Chen *et al.*, 2021), aunque los modelos posteriores han mejorado de forma considerable. Sin embargo, la comparación no es directa: HumanEval evalúa problemas algorítmicos autocontenidos a partir de una descripción abierta, mientras que este estudio evaluó la transformación de especificaciones ya estructuradas y ancladas a un sistema *legacy* con contexto arquitectónico explícito.

En ese sentido, el 85 % es más comparable con los estudios de migración de código *legacy*, donde se han reportado tasas de equivalencia funcional similares cuando el modelo dispone de contexto estructurado y de referencia al código original (Gandhi *et al.*, 2024). Esto sugiere que las especificaciones estructuradas y el código *legacy* como referencia cruzada pudieron contribuir al nivel de cumplimiento alcanzado, aunque el diseño del estudio no permite aislar su efecto frente a la capacidad del modelo.

Clasificación de incidencias

Durante la evaluación se identificaron 85 incidencias (Tabla 3). La categoría más frecuente fue la arquitectónica (35 casos, 41 %), seguida de la semántica (30 casos, 35 %) y la sintáctica (20 casos, 24 %). Estas 85 incidencias no deben sumarse a los 500 servicios ni compararse con los 75 componentes que requirieron iteración, porque un mismo servicio puede presentar una o más incidencias.

El predominio de las incidencias arquitectónicas es coherente con la evidencia reciente sobre las limitaciones de los modelos de lenguaje en tareas de diseño arquitectónico: un estudio empírico controlado encontró que su desempeño se degrada

de forma marcada al aumentar la complejidad arquitectónica del problema, incluso cuando el código generado es sintácticamente correcto (Hadjichristofi *et al.*, 2026).

Una explicación plausible es que las decisiones estructurales, como los límites de servicio, los patrones de comunicación o la separación de responsabilidades, dependen de un contexto que con frecuencia no está completamente explícito en una especificación funcional individual (Esposito *et al.*, 2026).

Este estudio no midió el tiempo de corrección por tipo de incidencia, por lo que no puede afirmar qué categoría es más costosa de resolver.

Utilidad arquitectónica

La evaluación arquitectónica clasificó 430 servicios (86 %) como útiles y 70 (14 %) como innecesarios. El puntaje de la rúbrica se relacionó de manera clara con la clasificación final (Tabla 4): el 99,6 % de los componentes con puntaje 5 fue útil y el 100 % de los de puntaje 0 fue innecesario, mientras que en el rango intermedio (puntaje 3) la clasificación fue menos determinista, con un 67,8 % de utilidad, lo que refleja los casos en que el equipo debió ponderar los criterios incumplidos. Entre los 70 servicios innecesarios, los criterios incumplidos con mayor frecuencia fueron la reutilización (61,4 %), la pertinencia funcional (60,0 %) y la necesidad en la arquitectura objetivo (55,7 %). La IFCR y la utilidad miden dimensiones distintas, por lo que sus valores no son intercambiables ni necesariamente coinciden en los mismos componentes. De los 425 servicios que cumplieron la especificación sin ajustes, 361 (84,9 %) fueron útiles y 64 (15,1 %) innecesarios; de los 75 que requirieron iteración, 69 (92,0 %) fueron útiles y 6 (8,0 %) innecesarios (Tabla 5). Así, 64 de los 70 servicios innecesarios (91,4 %) habían cumplido la especificación sin ajustes, y 69 de los 75 servicios iterados terminaron siendo valiosos. Un servicio puede, por tanto, satisfacer la especificación y resultar redundante para el arquitecto, mientras que otro puede requerir una iteración y terminar siendo útil, lo que respalda la necesidad de una evaluación multidimensional.

Tabla 3

Clasificación de las incidencias identificadas

Categoría	Número de incidencias	Porcentaje
Sintácticas	20	24 %
Semánticas	30	35 %
Arquitectónicas	35	41 %
Total	85	100 %

Fuente: elaboración propia (2026).

Tabla 4

Relación entre el puntaje de la rúbrica y la clasificación final

Puntaje	Innecesario	Útil	Total	% útil
0	5	0	5	0,0 %
1	15	1	16	6,2 %
2	23	1	24	4,2 %
3	19	40	59	67,8 %
4	7	149	156	95,5 %
5	1	239	240	99,6 %
Total	70	430	500	86,0 %

Fuente: elaboración propia (2026)

Tabla 5

Cruce entre cumplimiento funcional inicial y utilidad arquitectónica

Cumplimiento funcional inicial	Útil	Innecesario	Total
Cumplió sin ajustes	361 (84,9 %)	64 (15,1 %)	425
Requirió iteración	69 (92,0 %)	6 (8,0 %)	75
Total	430	70	500

Fuente: elaboración propia (2026) Porcentajes calculados por fila.

Productividad estimada

El tiempo estimado de desarrollo por componente fue de 4 días con el enfoque basado en inteligencia artificial generativa y de 10 días con el enfoque tradicional, lo que equivale a una diferencia de 6 días (ΔT), una reducción aproximada del 60 % y una razón de productividad temporal de 2,5 (RT). Ambos tiempos corresponden al ciclo completo de trabajo, que incluye el análisis y la especificación, las iteraciones y las pruebas, por lo que la comparación ya incorpora el costo de la corrección. Una reducción de esta magnitud es comparable a la reportada en un experimento aleatorizado con programadores profesionales, que encontró una disminución del 55,8 % en el tiempo de finalización de una tarea al usar GitHub Copilot frente a un grupo de control (Peng *et al.*, 2023).

Sin embargo, ese estudio midió tiempos reales en la escritura de código nuevo, mientras que aquí los tiempos son estimaciones derivadas de la *velocity*, por lo que la comparación es solo referencial. Una posible explicación es que parte del ahorro no provenga de escribir código más rápido, sino de reducir el esfuerzo de comprender el sistema existente e interpretar reglas de negocio ambiguas, que se desplaza hacia la formalización previa de las

especificaciones (Hou *et al.*, 2024; Sergeyuk *et al.*, 2025). Esta hipótesis no pudo contrastarse porque no se registraron tiempos por fase. Además, el resultado no implica que cualquier proceso asistido por inteligencia artificial sea 2,5 veces más rápido: corresponde al contexto, al tipo de componentes y a las condiciones del proyecto estudiado. Con factores de aceleración entre 1,5 y 3,0, la reducción estimada variaría entre 33 % y 67 %, de modo que la magnitud del beneficio depende del factor adoptado.

En conjunto, los resultados sugieren que la inteligencia artificial generativa es más útil como acelerador de la transformación de especificaciones que como generador autónomo de arquitectura. Esto tiene una implicación práctica: la inversión en especificaciones estructuradas y en validación arquitectónica parece ser una condición para obtener el beneficio y no un costo adicional, y la supervisión humana debería priorizarse en la evaluación de la coherencia arquitectónica más que en la revisión sintáctica, que puede automatizarse mediante compilación y pruebas. Además, una revisión sistemática de 395 artículos señala que la mayor parte de la evidencia proviene de tareas aisladas y no de procesos de transformación completos sobre sistemas reales (Hou *et al.*, 2024); este estudio aporta evidencia de un proceso completo, de la especificación al servicio, sobre 500 componentes de un sistema *legacy* real.

En cuanto a las limitaciones del estudio, primero, se realizó sobre un único sistema *legacy*, por lo que los resultados no pueden generalizarse a otros sistemas o dominios. Segundo, la muestra no fue aleatoria: se seleccionó por un criterio de complejidad (tres o más *Story Points*), lo que sitúa el estudio en un enfoque observacional y limita las conclusiones a servicios de complejidad relativamente mayor.

Tercero, la evaluación arquitectónica se cerró por consenso de un equipo de tres arquitectos y no mediante una evaluación ciega e independiente, por lo que no se pudo calcular un índice de concordancia entre evaluadores y no puede descartarse la influencia de la dinámica del consenso en la zona intermedia de puntaje; además, el criterio de necesidad en la arquitectura objetivo se solapa conceptualmente con la clasificación de utilidad.

Cuarto, los tiempos de ambos enfoques se estimaron a partir de la *velocity* del equipo aplicada a los *Story Points* de cada componente, y el factor de aceleración se obtuvo de los mismos 500 servicios desarrollados posteriormente con inteligencia artificial generativa, por lo que la comparación depende de un único factor promedio y no permite descartar efectos de aprendizaje: al tratarse de los mismos servicios, desarrollados primero de forma tradicional y después con inteligencia artificial generativa, el conocimiento previo de la funcionalidad puede haber reducido el tiempo del segundo enfoque. Además, los Story Points son estimaciones de expertos que pueden no reflejar con exactitud el esfuerzo real (Tawosi *et al.*, 2022). Como la variabilidad entre componentes está

determinada por su categoría de *Story Points*, no fue posible aplicar pruebas inferenciales ni calcular intervalos de confianza. Por ello, la comparación de 4 y 10 días debe leerse como una estimación de productividad y no como una medición con variabilidad muestral propia.

Se recomienda ampliar la evaluación a otros sistemas *legacy*, dominios y niveles de complejidad; comparar distintos modelos y configuraciones; registrar tiempos reales por fase para validar el modelo de productividad; y profundizar en la automatización de la evaluación arquitectónica y de la clasificación de incidencias (Kavitha *et al.*, 2025).

Para finalizar cabe destacar que los desafíos a los que se enfrentan las sociedades yla comunidad científica y académica son complejos; sin embargo, también se es testigo de un momento histórico en el que la colaboración y la innovación pueden transformar la realidad.

La globalización, la interconexión y la sostenibilidad no son meros conceptos, sino pilares que definen la época y que determinarán el legado que se dejará a las generaciones futuras (Ramírez, (2025).

Conclusiones

El objetivo del estudio fue evaluar un enfoque basado en inteligencia artificial generativa para transformar especificaciones funcionales de un sistema *legacy* real en servicios de software. Los resultados permiten concluir que el enfoque logra que la mayor parte de los servicios evaluados cumpla la especificación sin ajustes y que la gran mayoría resulte útil para la arquitectura objetivo, y que los componentes restantes se resuelven mediante iteración sobre la especificación. Las incidencias se concentran en el plano arquitectónico más que en el sintáctico, lo que indica que la validación técnica automatizada es necesaria pero insuficiente. En productividad, la estimación disponible apunta a una reducción sustancial del tiempo de desarrollo por componente, aunque, al tratarse de una estimación y no de una medición, este resultado debe tomarse como indicativo.

La principal conclusión es que la inteligencia artificial generativa puede emplearse como un mecanismo de transformación asistida, siempre que se integre con especificaciones estructuradas, validación funcional y técnica, evaluación arquitectónica e iteración.

Los resultados no respaldan una automatización autónoma, sino un proceso controlado en el que la generación automática y la supervisión especializada trabajan conjuntamente, con la supervisión humana concentrada en la coherencia arquitectónica. Asimismo, el estudio propone un marco de evaluación multidimensional que podría aplicarse en otros proyectos de modernización.

Agradecimientos

Este artículo se deriva de la tesis del autor en la Universidad de La Salle, Colombia.

Referencias

Bakar, H. K. A., Razali, R. and Jambari, D. I. (2019). Implementation phases in modernisation of legacy systems. *2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS)*, 1–6. https://doi.org/10.1109/ICRIIS48246.2019.9073628

Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde de Oliveira Pinto, H., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S. (…) Zaremba, W. (2021). *Evaluating large language models trained on code*. arXiv. https://doi.org/10.48550/arXiv.2107.03374

Crotty, J. and Horrocks, I. (2017). Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company. *Applied Computing and Informatics*, *13*(2), 175–183. https://doi.org/10.1016/j.aci.2016.12.001

Diggs, C., Doyle, M., Madan, A., Scott, S., Escamilla, E., Zimmer, J., Rothermel, K. J., Sanchez, D., Robinson, S., Zwyssig, T. and Thaker, S. (2024). *Leveraging LLMs for legacy code modernization: Challenges and opportunities for LLM-generated documentation*. arXiv. https://doi.org/10.48550/arXiv.2411.14971

Duarte, D. y Chacón, O. (2022). Inteligencia artificial: retos y desafíos de la ética laboral en la sociedad tecnológica, *Mundo Fesc*, vol 12 no. S3 pp. 265-280 de 2022

Esposito, M., Li, X., Moreschini, S., Ahmad, N., Cerny, T., Vaidhyanathan, K., Lenarduzzi, V. and Taibi, D. (2026). Generative AI for software architecture. Applications, challenges, and future directions. *Journal of Systems and Software*, *231*, 112607. https://doi.org/10.1016/j.jss.2025.112607

Etikan, I., Musa, S. A. and Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American Journal of Theoretical and Applied Statistics*, *5*(1), 1–4. https://doi.org/10.11648/j.ajtas.20160501.11

Fávero, L. F., de Almeida, N. R. and Affonso, F. J. (2025). A systematic mapping study on the modernization of legacy systems to microservice architecture. *Applied System Innovation*, *8*(4), 86. https://doi.org/10.3390/asi8040086

Gandhi, S., Patwardhan, M., Khatri, J., Vig, L. and Medicherla, R. K. (2024). Translation of low-resource COBOL to logically correct and readable Java leveraging high-resource Java refinement. *Proceedings of the 1st International Workshop on Large Language Models for Code*, 46–53. https://dl.acm.org/doi/10.1145/3643795.3648388

Gholami, M. F., Daneshgar, F., Beydoun, G. and Rabhi, F. (2017). Challenges in migrating legacy software systems to the cloud — an empirical study. *Information Systems*, *67*, 100–113. https://doi.org/10.1016/j.is.2017.03.008

Hadjichristofi, C., Tsilimigkounakis, M., Sotiropoulos, G. and Vescoukis, V. (2026). An empirical evaluation of large language models applying software architectural patterns. *AI*, *7*(6), 195. https://doi.org/10.3390/ai7060195

Hasan, M. H., Osman, M. H., Admodisastro, N. I. and Muhammad, M. S. (2023). Legacy systems to cloud migration: A review from the architectural perspective. *Journal of Systems and Software*, *202*, 111702. https://doi.org/10.1016/j.jss.2023.111702

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J. and Wang, H.

(2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, *33*(8), 220. https://doi.org/10.1145/3695988

Irani, Z., Abril, R. M., Weerakkody, V., Omar, A. and Sivarajah, U. (2023). The impact of legacy systems on digital transformation in European public administration: Lesson learned from a multi case analysis. *Government Information Quarterly*, *40*(1), 101784. https://doi.org/10.1016/j.giq.2022.101784

Kavitha, V., Gowri Laksshmi, K. S., Hemashree, P. and Shobana, R. (2025). Empowering software engineering automation through explainable AI. En *Advances in Computers*. Elsevier. https://doi.org/10.1016/bs.adcom.2025.07.009

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S. and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, *33*. https://doi.org/10.48550/arXiv.2005.11401

Margara, A. and Salvaneschi, G. (2018). On the semantics of distributed reactive programming: The cost of consistency. *IEEE Transactions on Software Engineering*, *44*(7), 689–711. https://doi.org/10.1109/TSE.2018.2833109

Nguyen-Duc, A., Cabrero-Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., Rafiq, U., Melegati, J., Guerra, E., Kemell, K.-K., Saari, M., Zhang, Z., Le, H., Quan, T. and Abrahamsson, P. (2023). *Generative artificial intelligence for software engineering – A research agenda*. arXiv. https://doi.org/10.48550/arXiv.2310.18648

Peng, S., Kalliamvakou, E., Cihon, P. and Demirer, M. (2023). *The impact of AI on developer productivity: Evidence from GitHub Copilot*. arXiv. https://arxiv.org/pdf/2302.06590

Pérez-Castillo, R., De Guzmán, I. G.-R. and Piattini, M. (2011). Knowledge discovery metamodel-ISO/IEC 19506: A standard to modernize legacy systems. *Computer Standards & Interfaces*, *33* (6), 519–532. https://doi.org/10.1016/j.csi.2011.02.007

Ramírez, A. (2025). Globalization, interconnection or sustainability. *Revista Temario Científico, 5* (1), 1–3. https://doi.org/10.47212/rtcAlinin.3.125.1

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8

Sandborn, P. A. and Prabhakar, V. J. (2015). The forecasting and impact of the loss of critical human skills necessary for supporting legacy systems. *IEEE Transactions on Engineering Management*, *62*(3), 361–371. https://doi.org/10.1109/TEM.2015.2438820

Sergeyuk, A., Golubev, Y., Bryksin, T. and Ahmed, I. (2025). Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward. *Information and Software Technology*, *178*, 107610. https://doi.org/10.1016/j.infsof.2024.107610

Tawosi, V., Moussa, R. and Sarro, F. (2022). On the relationship between story points and development effort in agile open-source software. *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '22)*. https://

doi.org/10.1145/3544902.3546238

Tawosi, V., Moussa, R. and Sarro, F. (2023). Agile effort estimation: Have we solved the problem yet? Insights from a replication study. *IEEE Transactions on Software Engineering*, *49*(4), 2677–2697. https://doi.org/10.1109/TSE.2022.3228739