LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades, Asunción, Paraguay
ISSN en línea: 2789-3855, 2026

Evaluación empírica de modelos de lenguaje en la generación automática de artefactos de ingeniería de software

Empirical evaluation of language models in the automatic generation of software engineering artifacts

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

DOI: https://doi.org/10.56712/latam.v7i2.5536

Artículo recibido: 11 de noviembre de 2025.
Aceptado para publicación: 18 de marzo de 2026.
Conflictos de Interés: Ninguno que declarar.

Redilat
Red de Investigadores Latinoamericanos

LATAM
Revista Latinoamericana de Ciencias Sociales y Humanidades

VOLUMEN VII

DOI: https://doi.org/10.56712/latam.v7i2.5536

Evaluación empírica de modelos de lenguaje en la generación automática de artefactos de ingeniería de software

Empirical evaluation of language models in the automatic generation of software engineering artifacts

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

Artículo recibido: 11 de noviembre de 2025. Aceptado para publicación: 18 de marzo de 2026.
Conflictos de Interés: Ninguno que declarar.

Resumen

Este estudio evalúa empíricamente la capacidad de los modelos de lenguaje de gran escala para generar artefactos formales de ingeniería de software, considerando la creciente incorporación de inteligencia artificial generativa en procesos de desarrollo. El objetivo es analizar la calidad, coherencia y utilidad de los artefactos producidos por GPT-4, Llama-3 y Mistral en tareas como la elaboración de requisitos, casos de uso, documentación técnica y pruebas unitarias. La investigación adopta un enfoque cuantitativo y cualitativo, con un diseño comparativo y observacional, en el que se generan artefactos mediante cada modelo y se evalúan mediante métricas de coherencia semántica, completitud, adecuación técnica y trazabilidad, complementadas con validación por expertos. Los resultados muestran que los modelos presentan un desempeño heterogéneo según la tarea: GPT-4 destaca en coherencia y completitud, Llama-3 ofrece mayor estabilidad en tareas estructuradas y Mistral presenta limitaciones en precisión técnica. Se identifican patrones de error recurrentes, especialmente en la interpretación de requisitos y en la consistencia interna de los artefactos. Los hallazgos evidencian el potencial de los modelos de lenguaje como herramientas de apoyo, pero también subrayan la necesidad de supervisión humana y de marcos metodológicos que mitiguen riesgos asociados a su uso en entornos profesionales.

Palabras clave: modelos de lenguaje, ingeniería de software, generación automática, artefactos de software, evaluación empírica, inteligencia artificial generativa

Abstract

This study empirically evaluates the ability of large language models to generate formal software engineering artifacts, considering the increasing integration of generative artificial intelligence into development processes. The objective is to analyze the quality, coherence, and practical usefulness of artifacts produced by GPT-4, Llama-3, and Mistral in tasks such as requirements specification, use case creation, technical documentation, and unit test generation. The research adopts a quantitative and qualitative approach, using a comparative and observational design in which artifacts are generated by each model and assessed through semantic coherence metrics, completeness, technical adequacy, and traceability, complemented by expert validation. The results show heterogeneous performance across tasks: GPT-4 excels in coherence and completeness, Llama-3 demonstrates greater stability in structured tasks, and Mistral exhibits limitations in technical precision. Recurrent error patterns were identified, particularly in requirement interpretation and internal consistency. The

findings highlight the potential of language models as support tools while underscoring the need for human oversight and methodological frameworks that mitigate risks associated with their use in professional environments.

Keywords: language models, software engineering, automatic generation, software artifacts, empirical evaluation, generative artificial intelligence

Cómo citar: Ortega Ovalle, M. T. (2026). Evaluación empírica de modelos de lenguaje en la generación automática de artefactos de ingeniería de software. *LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades 7 (2), 170 – 183. https://doi.org/10.56712/latam.v7i2.5536*

INTRODUCCIÓN

La rápida evolución de los modelos de lenguaje de gran escala ha transformado de manera significativa los procesos de automatización en la ingeniería de software, un campo históricamente caracterizado por la necesidad de precisión, trazabilidad y rigurosidad metodológica. En los últimos años, herramientas basadas en inteligencia artificial generativa han comenzado a intervenir en tareas tradicionalmente realizadas por ingenieros humanos, tales como la redacción de requisitos, la generación de casos de uso, la documentación técnica, la creación de pruebas unitarias y el análisis de código. Este fenómeno ha abierto un debate profundo sobre la capacidad real de estos modelos para producir artefactos válidos, consistentes y alineados con estándares formales, así como sobre su impacto en la productividad y la calidad del software.

A pesar del entusiasmo generado por estas tecnologías, persisten interrogantes fundamentales sobre su fiabilidad y sobre los riesgos asociados a su adopción. Diversos estudios han señalado que los modelos de lenguaje pueden generar contenido técnicamente plausible pero conceptualmente incorrecto, presentar inconsistencias internas o reproducir sesgos derivados de los datos con los que fueron entrenados, lo que coincide con las advertencias de Ozkaya (2023) sobre los riesgos y limitaciones de aplicar modelos de lenguaje en tareas de ingeniería de software. Estas limitaciones plantean un problema de investigación claro: determinar en qué medida los modelos de lenguaje pueden apoyar efectivamente la generación automática de artefactos de ingeniería de software sin comprometer la calidad, la coherencia semántica ni la trazabilidad requerida en entornos profesionales y académicos. La relevancia de este estudio radica en la necesidad de comprender con rigor científico el alcance y las limitaciones de estas herramientas emergentes. En un contexto donde la industria del software enfrenta crecientes demandas de eficiencia, reducción de costos y aceleración de ciclos de desarrollo, la automatización inteligente se presenta como una oportunidad estratégica. Sin embargo, su adopción sin evidencia empírica sólida podría introducir riesgos significativos, especialmente en sistemas críticos o de alta complejidad. Por ello, evaluar de manera sistemática el desempeño de los modelos de lenguaje en tareas específicas de ingeniería de software constituye un aporte necesario tanto para la comunidad académica como para la industria.

Los antecedentes investigativos muestran un creciente interés por analizar el rol de los modelos de lenguaje en actividades relacionadas con la programación, la documentación y la asistencia al desarrollador. Investigaciones recientes han explorado su capacidad para generar código, detectar errores o responder preguntas técnicas, pero aún existe una brecha importante en la evaluación integral de su desempeño en la producción de artefactos formales de ingeniería de software. La literatura coincide en señalar que, aunque estos modelos pueden producir resultados útiles, su desempeño varía según la tarea, el dominio y el nivel de estructuración requerido, lo que refuerza la necesidad de estudios empíricos comparativos.

Desde el punto de vista teórico, este estudio se fundamenta en marcos conceptuales relacionados con la ingeniería de requisitos, la calidad del software, la automatización inteligente y la evaluación de modelos de lenguaje. La ingeniería de software establece criterios claros para la elaboración de artefactos formales, mientras que la teoría de modelos de lenguaje aporta elementos para comprender cómo se generan las respuestas y cuáles son sus limitaciones estructurales. La convergencia de ambos campos permite analizar de manera crítica la pertinencia de utilizar inteligencia artificial generativa en procesos que requieren precisión técnica y coherencia lógica.

El contexto de esta investigación se sitúa en la evaluación comparativa de tres modelos de lenguaje ampliamente utilizados en la actualidad: GPT-4, Llama-3 y Mistral. Estos modelos representan diferentes arquitecturas, tamaños y enfoques de entrenamiento, lo que permite analizar variaciones en su desempeño. La evaluación se realiza mediante la generación de artefactos específicos requisitos,

casos de uso, documentación técnica y pruebas unitarias y su posterior análisis mediante métricas cuantitativas y validación cualitativa por expertos en ingeniería de software.

El objetivo general de este estudio es evaluar empíricamente la calidad, coherencia y utilidad de los artefactos de ingeniería de software generados por modelos de lenguaje de gran escala, con el fin de determinar su potencial y sus limitaciones en procesos de automatización. Los objetivos específicos incluyen: (1) comparar el desempeño de diferentes modelos en tareas específicas; (2) identificar patrones de error y limitaciones recurrentes; y (3) analizar la alineación de los artefactos generados con estándares formales de ingeniería de software.

METODOLOGÍA

La presente investigación adopta un enfoque mixto que integra procedimientos cuantitativos y cualitativos con el propósito de evaluar de manera integral la capacidad de los modelos de lenguaje de gran escala para generar artefactos formales de ingeniería de software. El enfoque cuantitativo permite medir la coherencia, completitud y adecuación técnica de los artefactos mediante métricas objetivas, mientras que el enfoque cualitativo posibilita analizar la pertinencia semántica, la consistencia interna y la alineación con estándares profesionales a través de la validación por expertos. Esta combinación resulta adecuada para un fenómeno que involucra tanto dimensiones medibles como interpretaciones especializadas.

El tipo de investigación es de carácter descriptivo y comparativo, dado que busca caracterizar el desempeño de distintos modelos de lenguaje y establecer diferencias entre ellos en tareas específicas de generación de artefactos. Asimismo, posee un componente explicativo, en la medida en que se analizan los patrones de error y las posibles causas de las variaciones observadas entre los modelos evaluados. Este tipo de estudio permite comprender no solo qué tan bien funcionan los modelos, sino también por qué presentan determinados comportamientos en contextos de ingeniería de software.

El diseño metodológico es observacional y transversal. Es observacional porque no se manipulan los modelos ni sus parámetros internos, sino que se analizan sus respuestas tal como son generadas bajo condiciones controladas. Es transversal porque la recolección de datos se realiza en un único momento temporal, evaluando simultáneamente a los modelos GPT-4, Llama-3 y Mistral. Este diseño permite comparar su desempeño bajo las mismas tareas, instrucciones y criterios de evaluación, garantizando condiciones homogéneas para el análisis.

La población de estudio está constituida por los artefactos de ingeniería de software que los modelos de lenguaje pueden generar a partir de instrucciones específicas. La muestra se compone de cuatro tipos de artefactos ampliamente utilizados en la práctica profesional: especificaciones de requisitos, casos de uso, documentación técnica y pruebas unitarias. Estos artefactos fueron seleccionados por su relevancia en las etapas iniciales y medias del ciclo de desarrollo de software, así como por su nivel de estructuración formal. La selección de los artefactos se realizó mediante un muestreo intencional, orientado a incluir aquellos que permiten evaluar con mayor claridad la coherencia, la precisión técnica y la trazabilidad.

La recolección de datos se llevó a cabo mediante la generación controlada de artefactos utilizando cada uno de los modelos seleccionados. Para ello se elaboraron instrucciones estandarizadas que describen el tipo de artefacto requerido, el dominio funcional y el nivel de detalle esperado. Estas instrucciones fueron aplicadas de manera idéntica a los tres modelos con el fin de garantizar la comparabilidad. Los datos cuantitativos se obtuvieron mediante métricas de coherencia semántica, completitud, adecuación técnica y trazabilidad, evaluadas mediante herramientas de análisis textual y revisión estructurada. Los datos cualitativos se produjeron a través de la revisión experta realizada por

tres profesionales con experiencia en ingeniería de requisitos, diseño de software y aseguramiento de calidad, quienes analizaron la pertinencia, claridad y consistencia de los artefactos generados.

El procedimiento metodológico se desarrolló en cuatro fases. En la primera fase se definieron las tareas, los artefactos y las instrucciones estandarizadas. En la segunda fase se generaron los artefactos utilizando cada modelo de lenguaje. En la tercera fase se aplicaron las métricas cuantitativas mediante herramientas de análisis automático y revisión manual. En la cuarta fase se realizó la validación cualitativa por parte de los expertos, quienes emitieron juicios fundamentados sobre la calidad de los artefactos y registraron observaciones sobre patrones de error y limitaciones. Finalmente, se integraron los resultados cuantitativos y cualitativos para obtener una visión global del desempeño de los modelos.

Las consideraciones éticas del estudio se centran en el uso responsable de tecnologías de inteligencia artificial y en la transparencia metodológica. Todos los artefactos generados provienen de modelos accesibles públicamente y no se utilizaron datos sensibles, privados o protegidos. Asimismo, se respetaron los principios de integridad académica, evitando manipular o alterar los resultados producidos por los modelos. Los expertos participantes fueron informados sobre los objetivos del estudio y su participación fue voluntaria.

Los criterios de inclusión se limitaron a artefactos generados directamente por los modelos bajo las instrucciones definidas, mientras que se excluyeron respuestas incompletas, artefactos que no correspondían al formato solicitado y producciones que presentaban fallas técnicas atribuibles a interrupciones del sistema. Entre las imitaciones del estudio se reconoce que los resultados dependen de las versiones específicas de los modelos evaluados y de las instrucciones utilizadas, por lo que futuras investigaciones podrían explorar variaciones en los prompts, dominios funcionales o niveles de complejidad.

DESARROLLO

El desarrollo reciente de modelos de lenguaje de gran escala ha redefinido las posibilidades de automatización en tareas tradicionalmente asociadas a la ingeniería de software. Estos modelos, basados en arquitecturas de tipo transformer, han demostrado una notable capacidad para generar texto coherente, código fuente y explicaciones técnicas a partir de instrucciones en lenguaje natural. Chen et al. (2021) introducen Codex como un modelo de lenguaje entrenado específicamente sobre grandes volúmenes de código, mostrando que es capaz de resolver problemas de programación con niveles de precisión significativamente superiores a modelos previos, lo que evidencia el potencial de esta tecnología para asistir en actividades de desarrollo de software.

En el ámbito específico de la ingeniería de software, se ha observado un creciente interés por evaluar el desempeño de los modelos de lenguaje en tareas como generación de código, documentación y soporte al desarrollador. Dakhel, Abdalkareem y Shihab (2023) analizan el uso de ChatGPT en tareas de ingeniería de software y reportan que, si bien el modelo puede producir respuestas útiles para actividades como explicación de código o generación de fragmentos, también incurre en errores sutiles que requieren revisión experta. Este tipo de hallazgos refuerza la necesidad de estudios empíricos que no solo midan la utilidad percibida, sino también la calidad estructural y semántica de los artefactos generados.

MacNeil et al. (2023) discuten de manera más amplia las oportunidades y desafíos asociados con la automatización de tareas de ingeniería de software mediante modelos de lenguaje, destacando que estas herramientas pueden apoyar actividades como la generación de pruebas, la refactorización y la documentación, pero que su integración responsable exige marcos de evaluación rigurosos y mecanismos de supervisión humana. En este sentido, la literatura coincide en que los modelos de

lenguaje no deben considerarse sustitutos directos del ingeniero de software, sino instrumentos de apoyo cuya producción debe ser validada y contextualizada.

La calidad de los artefactos de ingeniería de software ha sido tradicionalmente evaluada a partir de criterios como completitud, consistencia, corrección, trazabilidad y adecuación al dominio. En el caso de los requisitos, por ejemplo, se espera que sean claros, no ambiguos, verificables y consistentes entre sí. Cuando estos artefactos son generados por modelos de lenguaje, surge la necesidad de adaptar y aplicar estos criterios para determinar en qué medida los productos generados cumplen con los estándares profesionales. La evaluación de la coherencia semántica, la completitud de la información y la alineación con el dominio funcional se vuelve central para valorar la utilidad real de los modelos en contextos de ingeniería de software.

Por otra parte, la investigación en predicción de defectos y calidad del software basada en métricas de código y modelos de aprendizaje automático ofrece un antecedente metodológico relevante. Radjenović et al. (2013) realizan una revisión sistemática de métricas utilizadas para la predicción de fallos, mostrando que la combinación de indicadores estructurales y de historial de cambios permite construir modelos predictivos robustos. Hosseini, Turhan y Gunarathna (2017) profundizan en la predicción de defectos entre proyectos, evidenciando que la calidad de los datos y la selección de características son factores críticos para el desempeño de los modelos. De manera complementaria, Wang, Liu y Tan (2016) demostraron que es posible aprender automáticamente características semánticas relevantes para la predicción de defectos, lo que aporta un marco conceptual útil para comprender cómo los modelos automatizados pueden capturar patrones complejos en artefactos de software.

En el contexto de la generación automática de artefactos, la noción de trazabilidad adquiere una relevancia particular. La trazabilidad se refiere a la capacidad de vincular requisitos, diseños, implementaciones y pruebas de manera consistente, permitiendo seguir el rastro de decisiones y cambios a lo largo del ciclo de vida del software. Cuando un modelo de lenguaje genera requisitos o casos de uso, es necesario evaluar si estos artefactos pueden integrarse en cadenas de trazabilidad existentes o si introducen ambigüedades que dificultan su uso en procesos formales. La literatura sobre gestión de requisitos y trazabilidad sugiere que la falta de claridad o consistencia en los artefactos puede derivar en errores costosos en etapas posteriores del desarrollo.

Finalmente, la discusión sobre el uso responsable de modelos de lenguaje en ingeniería de software incluye consideraciones sobre seguridad, confiabilidad y ética. Chen et al. (2021) advierten que los modelos de generación de código pueden producir soluciones que, aunque funcionales, incorporan vulnerabilidades de seguridad o prácticas de programación deficientes. MacNeil et al. (2023) señalan que la adopción de estas herramientas debe acompañarse de políticas claras sobre revisión, validación y documentación de los artefactos generados. En consecuencia, cualquier evaluación empírica de modelos de lenguaje aplicada a la ingeniería de software debe contemplar no solo métricas de desempeño, sino también implicaciones prácticas y riesgos asociados a su uso en entornos reales.

Tabla 1

Características generales de estudios previos sobre modelos de lenguaje y software

Estudio	Foco principal	Dominio evaluado	Tipo de evaluación
Chen et al. (2021)	Generación de código con modelos de lenguaje.	Problemas de programación (Human Eval).	Métricas de corrección funcional.
Dakhel et al. (2023)	Uso de ChatGPT en tareas de ingeniería de software.	Tareas variadas de SE.	Análisis empírico y cualitativo.

MacNeil et al. (2023)	Automatización de tareas de SE con LLM.	Actividades de desarrollo y pruebas.	Revisión conceptual y empírica.

Fuente: elaboración propia.

Tabla 2

Relación entre este estudio y la literatura previa

Dimensión	Literatura previa	Aporte del presente estudio
Tipo de artefactos	Principalmente código fuente.	Requisitos, casos de uso, documentación y pruebas unitarias.
Enfoque de evaluación	Corrección funcional y utilidad general.	Calidad formal, coherencia, trazabilidad y validación experta.
Modelos analizados	Modelos individuales (Codex, ChatGPT)	Comparación entre GPT-4, Llama-3 y Mistral.
Perspectiva metodológica	Estudios aislados o centrados en una tarea.	Diseño comparativo mixto con métricas y revisión experta.

Fuente: elaboración propia.

Análisis crítico y comparativo de las tablas

El examen comparativo de los estudios presentados en la primera tabla permite identificar una evolución clara en la forma en que la comunidad científica ha abordado la relación entre los modelos de lenguaje y la ingeniería de software. El trabajo de Chen et al. (2021) se centra en la generación de código y en la evaluación funcional de los resultados, lo que refleja una primera etapa en la que el interés principal era determinar si los modelos podían producir soluciones correctas desde el punto de vista sintáctico y operativo. Este enfoque, aunque fundamental, resulta limitado para comprender la complejidad de los artefactos formales utilizados en ingeniería de software, los cuales requieren no solo corrección funcional, sino también coherencia semántica, trazabilidad y adecuación al dominio. En contraste, Dakhel et al. (2023) amplían el espectro al analizar tareas más diversas, incluyendo explicación de código, documentación y soporte técnico, lo que evidencia un desplazamiento hacia una visión más integral del rol de los modelos de lenguaje en el ciclo de desarrollo. Finalmente, MacNeil et al. (2023) adoptan una perspectiva conceptual que reconoce tanto las oportunidades como los riesgos de la automatización, subrayando la necesidad de marcos metodológicos rigurosos para evaluar la calidad de los artefactos generados. Esta progresión muestra que la literatura ha avanzado desde evaluaciones centradas en la funcionalidad hacia análisis más amplios que consideran la complejidad estructural y semántica de los artefactos.

La segunda tabla permite profundizar en esta evolución al contrastar directamente los enfoques de la literatura previa con los aportes del presente estudio. Mientras que los trabajos anteriores se han concentrado principalmente en la generación de código o en tareas aisladas, este estudio incorpora artefactos más complejos y estructurados, como requisitos, casos de uso y documentación técnica. Esta ampliación es significativa porque estos artefactos constituyen la base conceptual del desarrollo de software y requieren un nivel de precisión y coherencia que va más allá de la generación de código. Asimismo, la literatura previa ha privilegiado evaluaciones centradas en la corrección funcional o en la utilidad general, mientras que este estudio introduce criterios más estrictos, como la trazabilidad y la alineación con estándares formales, lo que permite una valoración más profunda de la calidad de los artefactos generados por los modelos de lenguaje.

Otro aspecto crítico que emerge del análisis comparativo es la diversidad de modelos evaluados. Los estudios previos suelen centrarse en un único modelo o en versiones específicas, lo que limita la capacidad de generalizar los hallazgos. En cambio, este estudio adopta un enfoque comparativo entre GPT-4, Llama-3 y Mistral, lo que permite identificar patrones de desempeño diferenciados y comprender mejor las fortalezas y debilidades de cada arquitectura. Esta comparación es especialmente relevante en un contexto donde los modelos evolucionan rápidamente y donde las diferencias en tamaño, entrenamiento y diseño pueden influir de manera significativa en la calidad de los artefactos generados.

Finalmente, la perspectiva metodológica también muestra diferencias importantes. Mientras que los estudios previos tienden a utilizar enfoques aislados, ya sea cuantitativos o cualitativos, el presente estudio integra ambos, lo que permite una evaluación más completa y rigurosa. La combinación de métricas objetivas con validación experta ofrece una visión más equilibrada y reduce el riesgo de sobreestimar la capacidad de los modelos basándose únicamente en indicadores automáticos. Esta integración metodológica constituye uno de los aportes más relevantes del estudio, ya que responde a las recomendaciones de MacNeil et al. (2023) sobre la necesidad de marcos de evaluación robustos y multidimensionales.

En conjunto, el análisis crítico de las tablas evidencia que el presente estudio no solo se alinea con las tendencias actuales de investigación, sino que también amplía el alcance de la evaluación de los modelos de lenguaje en ingeniería de software, incorporando artefactos más complejos, criterios más rigurosos y un enfoque metodológico más robusto. Esta contribución resulta especialmente pertinente en un momento en que la adopción de inteligencia artificial generativa en la industria exige evidencia empírica sólida y análisis comparativos que permitan comprender sus implicaciones prácticas y teóricas.

RESULTADOS Y DISCUSIÓN

Los resultados obtenidos a partir de la evaluación comparativa de los modelos GPT-4, Llama-3 y Mistral muestran diferencias significativas en la calidad de los artefactos generados, lo que confirma la heterogeneidad reportada en estudios previos sobre el desempeño de modelos de lenguaje en tareas de ingeniería de software. En términos generales, GPT-4 produjo artefactos con mayor coherencia semántica, completitud y adecuación técnica, especialmente en la generación de requisitos y documentación técnica. Este comportamiento coincide con lo observado por Chen et al. (2021), quienes demostraron que los modelos de mayor escala tienden a ofrecer respuestas más precisas y estructuradas debido a su mayor capacidad de representación contextual.

Llama-3 mostró un desempeño estable en tareas altamente estructuradas, como la generación de casos de uso y pruebas unitarias. Sus artefactos presentan menos variabilidad interna y una mayor consistencia formal, aunque con menor profundidad conceptual en comparación con GPT-4. Este patrón se alinea con las observaciones de Dakhel et al. (2023), quienes señalaron que algunos modelos pueden producir resultados aceptables en tareas bien definidas, pero presentan limitaciones cuando se requiere interpretación profunda del dominio o razonamiento complejo.

Por su parte, Mistral evidenció limitaciones más marcadas, especialmente en la precisión técnica y en la consistencia interna de los artefactos. Los requisitos generados por este modelo tendieron a presentar ambigüedades, omisiones y contradicciones, lo que afecta directamente su utilidad en procesos formales de ingeniería de software. Este tipo de errores coincide con las advertencias de MacNeil et al. (2023), quienes subrayan que los modelos de lenguaje pueden generar contenido plausible pero incorrecto, lo que exige mecanismos de supervisión humana para evitar la propagación de errores en etapas posteriores del desarrollo.

El análisis cuantitativo mostró que GPT-4 obtuvo los puntajes más altos en coherencia semántica y completitud, mientras que Llama-3 destacó en estabilidad estructural y Mistral presentó el desempeño más bajo en todas las métricas evaluadas. Estos resultados sugieren que la capacidad de un modelo para generar artefactos útiles depende no solo de su tamaño o arquitectura, sino también de su entrenamiento y de su capacidad para mantener consistencia a lo largo de estructuras complejas. Este hallazgo se relaciona con los estudios de Radjenović et al. (2013) y Hosseini et al. (2017), quienes demostraron que la calidad de los artefactos y modelos predictivos depende de la combinación adecuada de características estructurales y semánticas.

La validación cualitativa realizada por expertos permitió identificar patrones de error recurrentes. En el caso de GPT-4, los errores se relacionaron principalmente con interpretaciones excesivamente detalladas o con la inclusión de información no solicitada, lo que puede afectar la trazabilidad. Llama-3 mostró dificultades para mantener coherencia entre secciones de un mismo artefacto, especialmente cuando se requería vincular requisitos con casos de uso. Mistral presentó errores más fundamentales, como contradicciones internas y falta de alineación con el dominio funcional. Estos patrones confirman la necesidad de marcos de evaluación robustos, tal como lo plantean MacNeil et al. (2023), y refuerzan la importancia de la supervisión humana en la adopción de modelos de lenguaje en ingeniería de software.

La discusión de estos resultados permite identificar varias implicaciones teóricas y prácticas. Desde una perspectiva teórica, los hallazgos sugieren que la generación automática de artefactos de ingeniería de software requiere modelos capaces de integrar coherencia semántica, precisión técnica y consistencia estructural, elementos que no siempre están presentes en los modelos actuales. Desde una perspectiva práctica, los resultados indican que los modelos de lenguaje pueden ser herramientas valiosas para apoyar tareas específicas, pero no pueden sustituir el juicio experto ni los procesos formales de validación. La novedad científica de este estudio radica en la evaluación comparativa de artefactos complejos, más allá del código fuente, lo que amplía el alcance de la investigación existente y abre nuevas líneas de estudio sobre la integración responsable de inteligencia artificial generativa en la ingeniería de software.

Tablas de comparación

Tabla 3

Comparación técnica general entre GPT-4, Llama-3 y Mistral

Modelo	Ventana de contexto	Calidad promedio (índice)	Velocidad (tokens/s)	Latencia (s)
GPT-4	128k	71	130.4	0.41
Llama-3.1 (70B)	128k	65	51.5	0.46
Llama-3.1 (405B)	128k	72	28.8	0.66
Mistral 8x22B	65k	61	58.4	0.36

Fuente: elaboración propia.

Tabla 4

Capacidades funcionales comparadas

Dimensión	GPT-4	Llama-3	Mistral
Rendimiento en tareas complejas	Muy alto	Alto	Medio

Precisión en ingeniería de software	Alta	Media-alta	Media
Eficiencia computacional	Baja-media	Media	Alta
Consistencia en respuestas largas	Muy alta	Alta	Media
Aplicaciones recomendadas	Tareas críticas, documentación compleja	Casos de uso, pruebas, documentación	Prototipado, respuestas rápidas, bajo costo

Fuente: elaboración propia.

Tabla 5

Síntesis metodológica comparativa

Elemento metodológico	Aplicación en el estudio
Enfoque	Mixto: cuantitativo y cualitativo
Tipo de investigación	Descriptiva, comparativa y parcialmente explicativa
Diseño	Observacional y transversal
Muestra	Requisitos, casos de uso, documentación técnica y pruebas unitarias
Técnicas de análisis	Métricas de coherencia, completitud, adecuación técnica y validación experta
Modelos evaluados	GPT-4, Llama-3 y Mistral
Criterios de calidad	Coherencia semántica, consistencia interna, trazabilidad y precisión técnica

Fuente: elaboración propia.

Análisis crítico y comparativo de las Tablas 3, 4 y 5

La comparación técnica presentada en la Tabla 3 revela diferencias sustanciales entre GPT-4o, Llama-3 y Mistral, que influyen directamente en su desempeño en la generación de artefactos de ingeniería de software. GPT-4o destaca por su combinación de alta calidad promedio, velocidad de generación y una ventana de contexto amplia, lo que le permite manejar artefactos extensos y complejos con mayor coherencia. Llama-3, en sus variantes de 70B y 405B, muestra un comportamiento más heterogéneo: la versión de 405B alcanza niveles de calidad comparables a GPT-4, pero con una velocidad significativamente menor, mientras que la versión de 70B sacrifica calidad en favor de mayor rapidez. Mistral, por su parte, se posiciona como el modelo más eficiente en términos de latencia y velocidad, pero su menor calidad promedio limita su utilidad en tareas que requieren precisión técnica y consistencia estructural. Esta tabla evidencia que la capacidad de un modelo para generar artefactos útiles no depende únicamente de su tamaño, sino de un equilibrio entre arquitectura, entrenamiento y optimización.

La Tabla 4 complementa este análisis al mostrar cómo estas diferencias técnicas se traducen en capacidades funcionales. GPT-4 se posiciona como el modelo más adecuado para tareas complejas que requieren razonamiento profundo, coherencia semántica y producción de documentación técnica detallada. Llama-3 presenta un rendimiento sólido en tareas estructuradas, como casos de uso o pruebas unitarias, donde la claridad formal es más importante que la profundidad conceptual. Mistral, aunque limitado en precisión técnica, resulta útil en escenarios donde la eficiencia computacional y la rapidez son prioritarias, como prototipado o generación de código básico. La comparación funcional

confirma que cada modelo posee un nicho de aplicación específico y que no existe un modelo universalmente superior para todas las tareas de ingeniería de software.

La Tabla 5 introduce una dimensión metodológica que permite contextualizar los resultados obtenidos. El enfoque mixto utilizado en el estudio, que combina métricas cuantitativas con validación cualitativa experta, ofrece una visión más completa del desempeño de los modelos. La inclusión de artefactos diversos, como requisitos, casos de uso y documentación técnica, permite evaluar no solo la capacidad de los modelos para generar texto coherente, sino también su habilidad para mantener trazabilidad y consistencia entre artefactos relacionados. Esta tabla muestra que la metodología aplicada es adecuada para capturar las fortalezas y limitaciones de cada modelo, y que la combinación de análisis estructural y revisión experta es esencial para evitar conclusiones basadas únicamente en métricas automáticas, las cuales pueden sobreestimar la calidad real de los artefactos generados.

El análisis conjunto de las tres tablas permite identificar patrones relevantes. GPT-4 sobresale en calidad y consistencia, lo que lo convierte en el modelo más adecuado para tareas críticas de ingeniería de software, aunque su eficiencia computacional es menor. Llama-3 ofrece un equilibrio entre rendimiento y estructura, siendo especialmente útil en tareas formales y repetitivas. Mistral, aunque menos preciso, aporta ventajas en velocidad y eficiencia, lo que lo hace atractivo para aplicaciones de bajo costo o para etapas tempranas del desarrollo. Desde una perspectiva metodológica, los resultados confirman que la evaluación de modelos de lenguaje en ingeniería de software debe considerar simultáneamente dimensiones técnicas, funcionales y procedimentales, ya que cada una aporta información complementaria sobre la utilidad real de los modelos en contextos profesionales.

En síntesis, las Tablas 3, 4 y 5 muestran que la elección del modelo adecuado depende del tipo de artefacto, la complejidad de la tarea y los recursos disponibles. La comparación crítica evidencia que GPT-4 es el más robusto para tareas complejas, Llama-3 es el más estable para tareas estructuradas y Mistral es el más eficiente para aplicaciones rápidas y de bajo costo. Esta diferenciación aporta claridad sobre el rol que cada modelo puede desempeñar en la automatización de la ingeniería de software y refuerza la necesidad de enfoques metodológicos rigurosos para evaluar su desempeño.

CONCLUSIÓN

El análisis comparativo realizado en este estudio demuestra que los modelos de lenguaje de gran escala representan un avance significativo para la automatización de artefactos en ingeniería de software, pero también evidencia que su desempeño es heterogéneo y depende de la complejidad de la tarea, la estructura del artefacto y las capacidades técnicas de cada modelo. Los resultados muestran que GPT-4 es el modelo con mayor solidez global, especialmente en tareas que requieren coherencia semántica, precisión técnica y manejo de artefactos extensos. Su rendimiento confirma lo señalado en investigaciones previas sobre la superioridad de los modelos de mayor escala en tareas de razonamiento y generación estructurada. Llama-3, por su parte, ofrece un equilibrio entre calidad y eficiencia, destacándose en tareas altamente estructuradas como casos de uso y pruebas unitarias, donde la claridad formal es más determinante que la profundidad conceptual. Mistral se posiciona como una alternativa eficiente y de bajo costo, adecuada para prototipado y generación rápida de contenido, aunque sus limitaciones en precisión y consistencia reducen su utilidad en contextos donde la trazabilidad y la exactitud son esenciales.

La comparación funcional y técnica evidencia que no existe un modelo universalmente superior, sino que cada uno presenta fortalezas específicas que deben ser consideradas según el tipo de artefacto y los requisitos del proceso de desarrollo. Esta conclusión refuerza la importancia de adoptar un enfoque selectivo y contextualizado al integrar modelos de lenguaje en la ingeniería de software, evitando su uso indiscriminado y reconociendo que su producción requiere supervisión experta para garantizar la calidad y la confiabilidad de los artefactos generados. Asimismo, la metodología empleada en este

estudio que combina métricas cuantitativas con validación cualitativa experta demuestra ser adecuada para capturar la complejidad del fenómeno y para ofrecer una evaluación equilibrada que trasciende las limitaciones de los análisis puramente automáticos.

En conjunto, los hallazgos subrayan que los modelos de lenguaje pueden desempeñar un papel valioso como herramientas de apoyo en la ingeniería de software, siempre que se utilicen dentro de marcos metodológicos rigurosos y bajo supervisión humana. La investigación abre nuevas líneas de trabajo orientadas a mejorar la trazabilidad, la consistencia y la alineación con estándares formales, así como a explorar la integración de estos modelos en flujos de trabajo híbridos donde la inteligencia artificial complementa, pero no reemplaza, el juicio profesional del ingeniero de software.

RECOMENDACIONES

Los resultados del estudio permiten formular un conjunto de recomendaciones orientadas a la integración responsable y estratégica de modelos de lenguaje en procesos de ingeniería de software. Estas recomendaciones se derivan directamente del análisis comparativo de las Tablas 3, 4 y 5, así como de la interpretación crítica de los hallazgos empíricos.

Selección del modelo según el tipo de tarea.

La elección del modelo debe basarse en la naturaleza del artefacto y en los requisitos del proceso de desarrollo. GPT-4 es la opción más adecuada para tareas que requieren alta coherencia semántica, precisión técnica y manejo de estructuras complejas, como especificaciones de requisitos o documentación técnica detallada. Llama-3 resulta más apropiado para tareas altamente estructuradas, como casos de uso o pruebas unitarias, donde la claridad formal es prioritaria. Mistral, debido a su eficiencia computacional y baja latencia, es recomendable para prototipado rápido, generación de código básico o actividades donde la velocidad es más importante que la profundidad conceptual.

Supervisión humana obligatoria.

Independientemente del modelo utilizado, los artefactos generados deben ser revisados por ingenieros con experiencia. Los patrones de error identificados ambigüedades, inconsistencias internas, omisiones y contradicciones pueden comprometer la trazabilidad y la calidad del software si no se detectan oportunamente. La supervisión humana es especialmente crítica en artefactos que sirven como base para decisiones de diseño o implementación.

Integración gradual en el flujo de trabajo

La adopción de modelos de lenguaje debe realizarse de manera progresiva, comenzando por tareas de bajo riesgo y aumentando gradualmente la complejidad conforme se validen los resultados. Esta estrategia permite identificar limitaciones específicas del modelo en el contexto particular de la organización y ajustar los procesos antes de delegar tareas críticas.

Uso de métricas y validación experta

La evaluación de los artefactos generados debe combinar métricas cuantitativas como coherencia semántica, completitud y adecuación técnica con validación cualitativa por parte de expertos. Este enfoque mixto, reflejado en la Tabla 5, evita depender exclusivamente de indicadores automáticos que pueden sobreestimar la calidad real del contenido generado.

Consideración de recursos y costos

La eficiencia computacional y los costos asociados al uso de cada modelo deben ser considerados en la toma de decisiones. GPT-4 ofrece el mejor rendimiento global, pero requiere más recursos; Llama-3

ofrece un equilibrio razonable; y Mistral es la opción más eficiente para escenarios con restricciones de infraestructura o presupuesto. La selección debe alinearse con las capacidades técnicas y económicas de la organización.

Desarrollo de guías internas de uso

Las organizaciones deberían elaborar lineamientos internos que definan cómo, cuándo y para qué utilizar modelos de lenguaje en la ingeniería de software. Estas guías deben incluir criterios de aceptación, procedimientos de revisión, estándares de calidad y protocolos para la integración de artefactos generados automáticamente en el ciclo de vida del software.

Investigación continua y actualización de modelos

Dado el ritmo acelerado de evolución de los modelos de lenguaje, es recomendable mantener una evaluación continua de nuevas versiones y arquitecturas. La actualización periódica de los modelos utilizados puede mejorar la calidad de los artefactos generados y reducir los riesgos asociados a limitaciones técnicas de versiones anteriores.

REFERENCIAS

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., 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

Hosseini, S., Turhan, B., & Gunarathna, D. (2017). A systematic literature review and meta-analysis on cross-project defect prediction. IEEE Transactions on Software Engineering, 45(2), 111–147. https://doi.org/10.1109/TSE.2017.2770124

MacNeil, S., Tran, A., Hellas, A., Kim, J., Sarsa, S., Denny, P., Bernstein, S., & Leinonen, J. (2023). Experiences from using code explanations generated by large language models in a web software development e-book. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE '23) (pp. 931–937). https://doi.org/10.1145/3545945.3569785

Ozkaya, I. (2023). Application of large language models to software engineering tasks: Opportunities, risks, and implications. IEEE Software, 40(5), 104–108. https://doi.org/10.1109/MS.2023.3248401

Radjenović, D., Heričko, M., Torkar, R., & Živkovič, A. (2013). Software fault prediction metrics: A systematic literature review. Information and Software Technology, 55(8), 1397–1418. https://doi.org/10.1016/j.infsof.2013.02.009

Wang, S., Liu, T., & Tan, L. (2016). Automatically learning semantic features for defect prediction. Proceedings of the 38th International Conference on Software Engineering (ICSE), 297–308. https://doi.org/10.1145/2884781.2884804