

De la algoritmia clásica a la inteligencia generativa: la IA como nuevo paradigma en la ingeniería de software

From classical algorithms to generative intelligence: AI as a new paradigm in software engineering

De l'algorithme classique à l'intelligence générative : l'IA comme nouveau paradigme dans l'ingénierie logicielle.

Da algoritmia clássica à inteligência generativa: A IA como novo paradigma na engenharia de software.

Juan Rigoberto Castillo Serracín

Universidad de Panamá, Centro Regional Universitario de San Miguelito, Panamá.

juan.castillos@up.ac.pa

 <https://orcid.org/0009-0006-5821-7028>

Javier Miguel Gómez S.

Universidad de Panamá, Centro Regional Universitario de San Miguelito, Panamá.

javier.gomez@up.ac.pa

 <https://orcid.org/0009-0000-4583-5157>

DOI: <https://doi.org/10.48204/contacto.v6n1.11292>

Recibido: 18/12/2025

Aceptado: 28/03/2026

RESUMEN

La presente investigación analiza la ruptura epistemológica en la ingeniería de software provocada por la transición del paradigma determinista tradicional hacia modelos probabilísticos basados en Inteligencia Artificial Generativa. Mediante una revisión bibliográfica estructurada (2021-2026) y la triangulación de datos industriales, se examina la obsolescencia de la codificación manual en aplicaciones de propósito general, contrastada con su vigencia irreductible en sistemas críticos de seguridad (Safety-Critical Systems). Los resultados evidencian que el modelo asistido actual incrementa la productividad, pero introduce riesgos significativos de vulnerabilidad y deuda técnica debido al sesgo de automatización y la limitación de contexto. Se propone un nuevo Modelo de Ingeniería Generativa Sistémica fundamentado en ventanas de contexto masivo y razonamiento multimodal, concluyendo que el rol del ingeniero debe evolucionar ineludiblemente de la escritura sintáctica hacia la arquitectura y auditoría de sistemas estocásticos bajo un esquema de supervisión humana obligatoria.

Palabras clave: Ingeniería de Software, Inteligencia Artificial Generativa, Ciberseguridad, Auditoría de Sistemas.

ABSTRACT

This research analyzes the epistemological rupture in software engineering caused by the transition from the traditional deterministic paradigm towards probabilistic models based on Generative Artificial Intelligence. Through a structured bibliographic review (2021-2026) and triangulation of industrial data, the obsolescence of manual coding in general-purpose applications is examined, contrasted with its irreducible validity in Safety-Critical Systems. The results evidence that the current assisted model increases productivity but introduces significant security risks and technical debt due to automation bias and context limitations. A new Systemic Generative Engineering Model is proposed, based on massive context windows and multimodal reasoning, concluding that the engineer's role must inescapably evolve from syntactic writing towards the architecture and auditing of stochastic systems under a mandatory human-in-the-loop framework.

Keywords: Software Engineering, Generative Artificial Intelligence, Cybersecurity, Systems Auditing.

RÉSUMÉ

La présente recherche analyse la rupture épistémologique en génie logiciel provoquée par la transition du paradigme déterministe traditionnel vers des modèles probabilistes fondés sur l'intelligence artificielle générative. Au moyen d'une revue bibliographique structurée (2021-2026) et d'une triangulation de données industrielles, on examine l'obsolescence du codage manuel dans les applications à usage général, mise en contraste avec sa pertinence irréductible dans les systèmes critiques pour la sécurité (*Safety-Critical Systems*). Les résultats montrent que le modèle assisté actuel accroît la productivité, mais introduit des risques significatifs de vulnérabilité et de dette technique en raison du biais d'automatisation et de la limitation du contexte. Un nouveau modèle d'ingénierie générative systémique est proposé, reposant sur des fenêtres de contexte massif et un raisonnement multimodal, et l'on conclut que le rôle de l'ingénieur doit évoluer inéluctablement de l'écriture syntaxique vers l'architecture et l'audit de systèmes stochastiques dans un cadre de supervision humaine obligatoire.

Mots-clés : Génie logiciel, Intelligence artificielle générative, Cybersécurité, Audit de systèmes.

RESUMO

A presente investigação analisa a ruptura epistemológica na engenharia de software provocada pela transição do paradigma determinista tradicional para modelos probabilísticos baseados em Inteligência Artificial Generativa. Mediante uma revisão bibliográfica estruturada (2021-2026) e a triangulação de dados industriais, examina-se a obsolescência da codificação manual em aplicações de propósito geral, contrastada com sua vigência irreduzível em sistemas críticos de segurança (*Safety-Critical Systems*). Os resultados evidenciam que o modelo assistido atual aumenta a produtividade, mas introduz riscos significativos de vulnerabilidade e dívida técnica devido ao viés de automação e à limitação de contexto. Propõe-se um novo Modelo de Engenharia Generativa Sistêmica fundamentado em janelas de contexto massivo e raciocínio multimodal, concluindo que o papel do engenheiro deve evoluir inevitavelmente da escrita sintática para a arquitetura e auditoria de sistemas estocásticos sob um esquema de supervisão humana obrigatória.

Palavras-chave: Engenharia de Software, Inteligência Artificial Generativa, Cibersegurança, Auditoria de Sistemas.

Introducción

En la última década, la ingeniería de software ha iniciado una metamorfosis irreversible, transitando de un paradigma determinista y reduccionista, donde la creación de sistemas dependía exclusivamente de la capacidad humana para codificar reglas explícitas, hacia ecosistemas de desarrollo basados en la probabilidad y la generación sintética. Este enfoque tradicional, denominado Software 1.0, exigía que el programador descompusiera problemas complejos en secuencias lógicas de instrucciones precisas. Sin embargo, la irrupción de los Grandes Modelos de Lenguaje (LLM) y las arquitecturas *Transformers* ha precipitado una ruptura epistemológica en la disciplina, desplazando el foco desde la escritura de sintaxis hacia el diseño de objetivos y la curaduría estratégica de datos (Vaswani et al., 2017).

Esta transición hacia el Software 2.0 ha transformado el Ciclo de Vida del Desarrollo de Software (SDLC) en un ecosistema híbrido donde la barrera entre la especificación de requisitos y la implementación técnica se ha diluido. Investigaciones lideradas por GitHub y Microsoft sugieren que el uso de asistentes de IA puede aumentar la velocidad de codificación hasta en un 55%, lo que redefine la productividad en el sector (Peng et al., 2023). No obstante, esta aceleración ha disuelto las fronteras de la fiabilidad algorítmica, introduciendo una vulnerabilidad sistémica: la naturaleza estocástica de los modelos generativos. A diferencia del código manual, donde el error es rastreable a una decisión humana, el código sintético introduce "alucinaciones lógicas" y debilidades de seguridad que a menudo pasan desapercibidas. Estudios empíricos han demostrado que los desarrolladores que utilizan asistentes de IA tienden a escribir código menos seguro y, paradójicamente, muestran una mayor confianza en sus resultados, un fenómeno conocido como sesgo de automatización (Perry et al., 2023).

Esta evolución tecnológica ha exacerbado una vulnerabilidad crítica: la asimetría entre la velocidad de generación de código y la capacidad humana de auditoría semántica. La industria se enfrenta a una paradoja donde la productividad aumenta, pero la comprensibilidad y mantenibilidad del software disminuyen debido a la acumulación de una deuda técnica cognitiva. Como señalan Mozannar et al. (2024) en su análisis sobre el comportamiento del usuario en programación asistida, el costo de revisión y corrección de errores sutiles introducidos por la IA puede, en ciertos casos, neutralizar las ganancias de velocidad iniciales. En este contexto, los paradigmas de enseñanza y aseguramiento de calidad enfocados en la sintaxis manual resultan obsoletos. La ingeniería de software debe evolucionar, por lo tanto, desde una práctica de construcción artesanal hacia una disciplina de orquestación y auditoría de sistemas estocásticos (Knight, 2002).

La presente investigación se justifica por la necesidad urgente de formalizar este nuevo paradigma, articulando cómo los principios clásicos de modularidad y abstracción deben reinterpretarse. El problema central radica en la obsolescencia de las métricas de desarrollo actuales frente a la opacidad de los

algoritmos generativos; las organizaciones operan bajo una falsa premisa de exactitud algorítmica, careciendo de marcos estandarizados para validar la fiabilidad del código sintético.

A partir de esta problemática, surge la siguiente interrogante que guía la investigación: ¿De qué manera puede la ingeniería de software integrar la naturaleza probabilística de la IA generativa sin comprometer el determinismo y la seguridad exigidos en sistemas de misión crítica?

Para afrontar esta complejidad, el estudio tiene como objetivo general analizar la transición desde la algoritmia clásica hacia la programación generativa, proponiendo un Modelo de Ingeniería Generativa Sistémica fundamentado en el razonamiento multimodal y la supervisión humana obligatoria (*Human-in-the-loop*). Bajo esta perspectiva, se sostiene la hipótesis de trabajo de que la sostenibilidad del desarrollo futuro no dependerá de la capacidad de escritura de código, sino de la evolución del ingeniero hacia un arquitecto de sistemas probabilísticos, capaz de gestionar la incertidumbre de los modelos priorizando la verificación semántica y la integridad del diseño sobre la simple implementación sintáctica.

Metodología y materiales

Diseño del Estudio

Para el desarrollo de la presente investigación se adoptó un enfoque cualitativo con un diseño no experimental de carácter documental y bibliográfico. El estudio se define por un alcance descriptivo-analítico y propositivo, orientado a articular la transición ontológica desde la programación manual determinista hacia la ingeniería de software asistida por inteligencia artificial (IA).

Participantes (Fuentes Documentales)

Al tratarse de un estudio bibliográfico, los "participantes" corresponden a una muestra final de 18 fuentes primarias y reportes de alta relevancia estratégica. Esta muestra integra literatura científica indexada, marcos de estandarización técnica (como ISO/IEC 42001 y OWASP Top 10 for LLM) y reportes industriales estratégicos (GitHub Octoverse, Stack Overflow Developer Survey).

Materiales e instrumentos

El principal instrumento fue la ejecución de ecuaciones de búsqueda booleanas diseñadas para maximizar la exhaustividad y precisión técnica. Se emplearon cadenas de consulta en repositorios digitales de alto rigor como IEEE Xplore, ACM Digital Library, Scopus y Google Scholar, utilizando términos como: ("*Generative AI*" OR "*Large Language Models*") AND ("*Software Engineering*" OR "*SDLC*") AND ("*Cybersecurity*" OR "*Technical Debt*").

Procedimiento

El estudio se operacionalizó mediante una Revisión Bibliográfica Estructurada (RBE) acotada al

periodo 2021-2026. El proceso sistemático incluyó cuatro fases: a) Identificación: Localización de 278 registros iniciales; b) Cribado (Screening): Evaluación de títulos y resúmenes para descartar literatura fuera de foco; c) Elegibilidad: Revisión a texto completo de documentos con validación empírica; y d) Inclusión: Consolidación de la muestra final de 18 fuentes.

El horizonte temporal se acotó estrictamente al periodo 2021-2026, intervalo seleccionado deliberadamente por coincidir con la disrupción masiva de arquitecturas basadas en *Transformers* como Codex, GPT-4 y Llama 3. El proceso de selección se estructuró sistemáticamente en cuatro fases consecutivas: inició con la identificación mediante la localización de 278 registros en las bases de datos indexadas mencionadas, seguida de un riguroso cribado (*screening*) donde se evaluaron títulos y resúmenes para descartar literatura redundante o fuera de foco. Posteriormente, se procedió a la fase de elegibilidad, consistente en una revisión a texto completo de aquellos documentos que presentaran validación empírica o modelos de gestión contrastables, para culminar con la inclusión y consolidación de una muestra final de 18 fuentes primarias y reportes de alta relevancia estratégica. Esta muestra integra no solo literatura científica, sino también marcos de estandarización técnica, como ISO/IEC 42001 y OWASP Top 10 for LLM y reportes estratégicos de la industria, incluyendo GitHub Octoverse y Stack Overflow Developer Survey, garantizando así una visión integral del fenómeno.

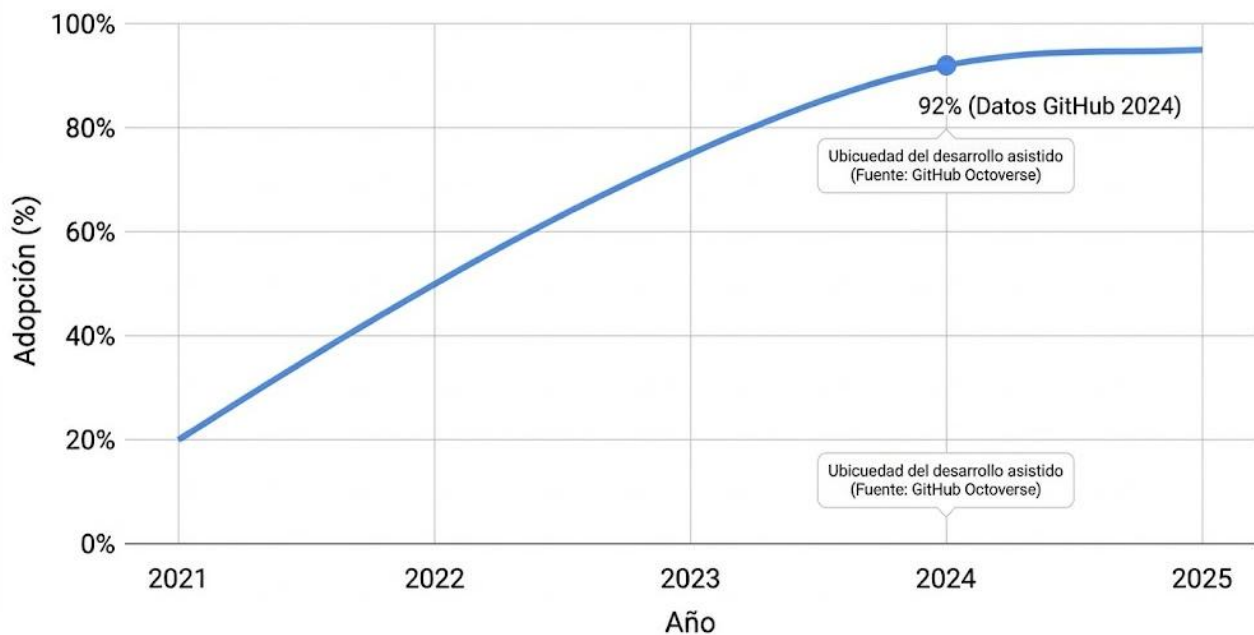
Para garantizar la validez de constructo, se aplicó una triangulación de fuentes heterogéneas, contrastando los hallazgos empíricos de la academia con la visión pragmática de la industria global. Esta convergencia permitió examinar las limitaciones del ciclo de vida de desarrollo de software (SDLC) tradicional frente a la incertidumbre estocástica de los modelos generativos. La calidad de la evidencia se aseguró priorizando fuentes sometidas a revisión por pares (*peer-review*) y con un alto factor de impacto, descartando estudios puramente algorítmicos que carecieran de validación operativa o impacto demostrable en la estrategia de ingeniería.

Mediante un enfoque hermenéutico, se validó la pertinencia de integrar la verificación formal y la ingeniería de *prompts* como competencias nucleares, redefiniendo el rol del programador de un escritor de sintaxis a un auditor de sistemas probabilísticos. Finalmente, el análisis se complementó con una revisión técnica dirigida de las capacidades de herramientas líderes, GitHub Copilot, Gemini Code Assist y ChatGPT, para evaluar sus controles de seguridad y calidad. Para evitar el sesgo de tecno-optimismo, el estudio confrontó críticamente los incrementos de productividad reportados (hasta un 55%) con los riesgos sistémicos de alucinaciones algorítmicas y el aumento de la deuda técnica en entornos de producción empresarial.

Resultados

El análisis documental y la triangulación de datos técnicos de la industria confirman la hipótesis de trabajo planteada: la ingeniería de software ha superado el umbral de la programación manual exclusiva. La capacidad de generación de código mediante IA no es una herramienta auxiliar, sino un cambio estructural que redefine la competencia técnica, desplazando el valor desde la sintaxis hacia la semántica y la verificación.

Figura 1
Curva de adopción de asistentes de codificación basados en IA (2021-2025).



Elaboración propia basada en datos de GitHub (2024).

Como se ilustra en la Figura 1, la trayectoria de crecimiento en la integración de herramientas de IA generativa dentro de los flujos de trabajo ha sido exponencial durante el último lustro. Al triangular estos hallazgos con el reporte *The State of the Octoverse* de GitHub (2024), se identifica un punto de inflexión crítico en 2024, donde la adopción alcanzó el 92% de la población de desarrolladores. Este fenómeno marca el fin de la etapa de innovación temprana para consolidar un estándar industrial ubicuo, validando el desplazamiento definitivo del paradigma de programación manual hacia un modelo híbrido asistido.

Esta tendencia demuestra que la utilización de asistentes de codificación ha trascendido su carácter experimental para instituirse como una competencia nuclear en la ingeniería moderna. Al respecto, el Manifiesto de Copenhague (Russo et al., 2024) establece que la disciplina ha ingresado en una fase donde la IA generativa no es un reemplazo, sino una extensión sociotécnica del ingeniero, cuya pericia ahora reside en la orquestación humana de sistemas inteligentes y en la garantía ética de la calidad del software. En este nuevo escenario, el entorno operativo evoluciona de un entorno de desarrollo integrado (IDE) pasivo hacia un ecosistema predictivo, transformando ineludiblemente el rol del ingeniero: de un redactor de código sintáctico a un supervisor y arquitecto de sistemas inteligentes.

Bajo este paradigma, la capacidad de generación de código no se percibe como una herramienta auxiliar, sino como un cambio estructural que desplaza el valor profesional desde la escritura de sintaxis hacia la semántica y la verificación crítica. Como sostienen Lakhani y Sull (2024), la IA está democratizando la pericia técnica y obligando a los líderes a repensar la estructura de los equipos, donde

el ingeniero actúa ahora como un 'compañero cibernético' encargado de la orquestación estratégica. En este sentido, investigaciones de IBM Security (2025) confirman que el uso de modelos de lenguaje a gran escala (LLM) ha reducido la barrera de entrada para la ejecución de tareas complejas, lo que exige que el profesional humano se enfoque en la resiliencia cognitiva y el análisis de autenticidad más que en la construcción manual de scripts.

Los resultados se estructuran en torno a los tres pilares que definen este cambio de paradigma hacia la ingeniería generativa:

1. La Erosión de la Sintaxis y el Ascenso de la Ingeniería de Prompts

El análisis confirma que las organizaciones están abandonando la valoración de la memorización sintáctica en favor de la capacidad de abstracción. En un entorno donde modelos como GPT-4 o Claude 3.5 pueden generar scripts funcionales en segundos, la distinción entre un desarrollador junior y uno senior ya no radica en saber escribir el código, sino en saber qué pedirle al modelo (Prompt Engineering).

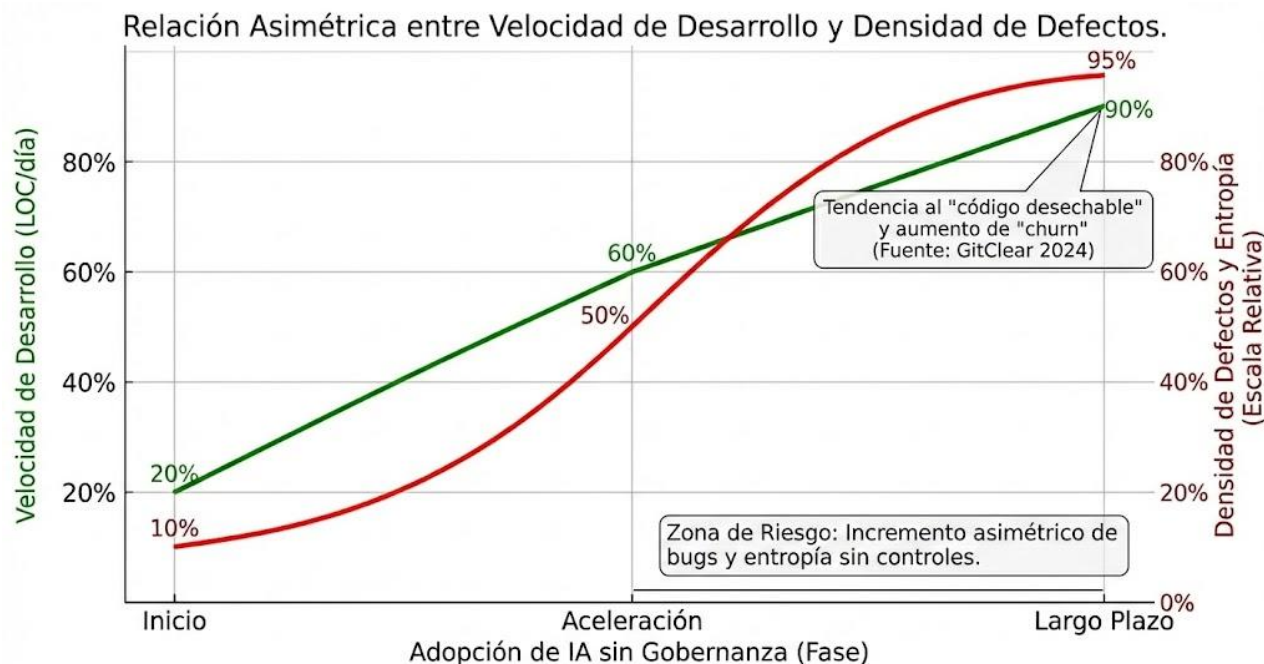
La revisión de la literatura técnica establece que el código fuente se está convirtiendo en un artefacto transitorio, un lenguaje intermedio entre la intención humana y el ejecutable de la máquina. Esto valida la necesidad de implementar metodologías de diseño declarativo, donde el ingeniero define los límites y restricciones del sistema, y la IA rellena la implementación técnica.

2. La Paradoja de la Productividad y la Deuda Técnica Cognitiva

Un hallazgo crítico de esta investigación es la correlación inversa entre la velocidad de generación de código y la profundidad de la comprensión semántica por parte del ingeniero.

Como se evidencia en la Figura 2, esta relación asimétrica plantea un desafío estructural para la sostenibilidad del software moderno. Si bien estudios de McKinsey (2023) revelan que la IA generativa puede acelerar la documentación y la generación de código hasta en un 55%, esta eficiencia operativa introduce una Deuda Técnica Cognitiva: el código se produce a una cadencia que supera la capacidad de auditoría y verificación del cerebro humano en tiempo real.

Figura 2.
Relación Asimétrica entre Velocidad de Desarrollo y Densidad de Defectos.



Elaboración propia basada en datos de GitClear (2024).

La validación empírica derivada de esta tendencia sugiere que, aunque la velocidad de construcción aumenta, la introducción de errores sutiles y vulnerabilidades crece exponencialmente en ausencia de controles rigurosos. El análisis de tendencias basado en GitClear (2024) indica un incremento sustancial en el "código desechable" y en la duplicación de bloques lógicos (*churn*), lo que demuestra que la adopción de inteligencia artificial sin una gobernanza adecuada incrementa la entropía del sistema.

Frente a esta problemática, la discusión evidencia una divergencia tecnológica clave en el rol de la ventana de contexto. Mientras que las herramientas estándar fragmentan el conocimiento del sistema, modelos de contexto masivo como Gemini 1.5 Pro permiten integrar repositorios enteros en la memoria de trabajo de la IA. Esta capacidad técnica mitiga la ceguera arquitectónica identificada en modelos previos, permitiendo refactorizaciones sistémicas que antes eran inalcanzables. En consecuencia, la transición del autocompletado de líneas hacia una verdadera Arquitectura Asistida se posiciona como el mecanismo indispensable para neutralizar la deuda técnica y garantizar la coherencia global de los sistemas estocásticos.

3. Inseguridad Estocástica y Alucinaciones Algorítmicas

El tercer pilar identificado en esta transición es la tensión inherente entre la naturaleza probabilística de los modelos de lenguaje (LLM) y el determinismo técnico exigido por el software seguro. La investigación documental destaca que la Alucinación de Paquetes y la invención de dependencias

inexistentes representan nuevos vectores de ataque críticos. A diferencia del error lógico humano, la IA puede generar llamadas a librerías vulnerables o ficticias basadas en patrones estadísticos obsoletos. Esta realidad exige que la seguridad se desplace radicalmente hacia la izquierda (*Shift Left*), integrando el escaneo de vulnerabilidades en la fase misma de generación y no solo durante el despliegue.

Figura 3

Nuevo Ciclo de Vida de Desarrollo de Software (SDLC) Híbrido.



Elaboración propia basada en los principios de "Secure by Design" (CISA, 2023) y marcos de gobernanza para LLMs.

El diagrama esquematiza el flujo de trabajo propuesto en esta investigación para un ciclo de vida de desarrollo de software híbrido. El proceso trasciende la ejecución lineal tradicional al introducir un paso crítico y obligatorio: la Auditoría Semántica y de Seguridad. Esta fase no es una simple revisión técnica, sino un proceso de validación cruzada que combina la experticia humana con herramientas de análisis estático (SAST) y dinámico. Este bucle de retroalimentación (Human-in-the-loop) se establece como la única barrera efectiva contra la degradación de la calidad y la infiltración de alucinaciones algorítmicas en entornos de producción. Bajo este esquema, la responsabilidad del ingeniero evoluciona de la producción de líneas hacia la certificación de la integridad del sistema, garantizando que el código sintético cumpla con los estándares de seguridad crítica antes de su integración final.

Análisis Comparativo de Modelos de Desarrollo

Para contrastar la evolución teórica con la praxis técnica, la presente investigación analiza tres enfoques paradigmáticos que coexisten en la industria actual. Esta comparación permite identificar no solo las ganancias en eficiencia, sino las barreras normativas que definen el alcance de la Inteligencia Artificial.

Caso 1: El Modelo Determinista y la Barrera de la Seguridad Crítica

El paradigma determinista, base de la programación tradicional, se fundamenta en la construcción imperativa y explícita de la lógica, donde el ingeniero asume la carga cognitiva total de traducir requisitos de negocio a sintaxis formal. Este modelo exige un dominio memorístico de librerías y lenguajes, obligando a la gestión manual de la memoria y estructuras de control flujo a flujo. Bajo este esquema, la calidad del software es directamente proporcional a la capacidad de atención sostenida del humano, operando bajo la premisa de que la funcionalidad no existe si no está codificada explícitamente.

Sin embargo, este enfoque resulta insostenible ante la complejidad ciclomática de los sistemas modernos debido a la fragilidad humana intrínseca. La literatura técnica estima una tasa de defectos promedio de 15 a 50 errores por cada 1,000 líneas de código (KLOC) en desarrollos manuales industriales. Asimismo, la carga cognitiva requerida ralentiza drásticamente el *time-to-market*; estudios recientes indican que el desarrollo puramente manual es entre un 30% y un 50% más lento en tareas repetitivas frente a flujos asistidos por IA, generando cuellos de botella críticos en la entrega de valor.

A pesar de que este paradigma se considera técnicamente obsoleto para el desarrollo web y de propósito general, con una adopción de IA que alcanzó el 92% en 2024 y proyecciones de Gartner que estiman que el 40% del código empresarial será sintético para 2025, el modelo determinista persiste como un estándar irreductible en sectores regulados. Esta persistencia responde a una incompatibilidad ontológica fundamental entre la naturaleza estocástica de los modelos de lenguaje (LLM) y las exigencias de las normativas de seguridad crítica.

Mientras las aplicaciones de propósito general toleran fallos estocásticos, el ecosistema de los Sistemas Críticos de Seguridad (*Safety-Critical Systems*) y la computación embebida exigen una trazabilidad y un determinismo que la naturaleza de caja negra de la IA actual no puede certificar. En este dominio, el modelo manual se mantiene como la única vía robusta para la gestión de riesgos; lenguajes como C y C++ conservan una dominancia técnica absoluta en sectores como la aviónica y dispositivos médicos. En estos escenarios, el cumplimiento de normativas como ISO 26262 (Automotriz), DO-178C (Aviónica) e IEC 62304 (Dispositivos Médicos) no es negociable, requiriendo un control manual de la gestión de memoria y una latencia cero que los modelos probabilísticos actuales son incapaces de garantizar. Por consiguiente, el paradigma determinista marca la frontera ontológica que la Inteligencia Artificial generativa aún no ha logrado traspasar.

Análisis de la Bifurcación Paradigmática en Sistemas de Misión Crítica

La transición hacia la inteligencia artificial generativa no es uniforme en todas las dimensiones de la ingeniería. La persistencia del modelo determinista en sectores regulados evidencia una barrera técnica y normativa que los modelos probabilísticos aún no han logrado sortear, como se detalla en el análisis comparativo por sector industrial que se presenta a continuación:

Tabla 1

Dominancia del Modelo Determinista (C/C++) en Sectores Críticos Regulados

Sector Industrial	Estándar de Seguridad (Normativa)	Requisito de Determinismo	Por qué la IA Generativa es Insuficiente hoy
Automotriz	ISO 26262 (ASIL D)	Gestión estricta de memoria sin Garbage Collection.	La IA no garantiza latencia cero ni trazabilidad línea por línea.
Aviónica	DO-178C (Nivel A)	Cobertura de código del 100% y MC/DC.	El código "alucinado" viola la certificación de vuelo.
Dispositivos Médicos	IEC 62304 (Clase C)	Prevención de fallos fatales en tiempo real.	La gestión de riesgos exige validación formal, no estadística.
Industrial / IoT	IEC 61508	Fiabilidad funcional a largo plazo (>20 años).	Obsolescencia rápida de modelos de IA vs. estabilidad de C/C++.

Elaboración propia a partir de los requisitos de cumplimiento de las normativas de seguridad funcional ISO 26262 (Automotriz), DO-178C (Aviación) e IEC 62304 (Software Médico).

Como se evidencia en la Tabla 1, la resistencia a abandonar el modelo determinista en sectores de alta criticidad no responde a una inercia cultural, sino a una incompatibilidad ontológica fundamental entre la arquitectura de los Grandes Modelos de Lenguaje (LLM) y la normativa de seguridad vigente. Los estándares internacionales exigen un rigor de trazabilidad donde cada instrucción de bajo nivel debe ser vinculada unívocamente a un requisito funcional y a un vector de prueba específico.

La naturaleza de la IA Generativa, al fundamentarse en principios estocásticos, introduce una opacidad algorítmica que rompe esta cadena de trazabilidad lineal. En consecuencia, mientras no se consoliden marcos de IA Explicable (XAI) y métodos de Verificación Formal capaces de certificar matemáticamente la salida de redes neuronales, la ingeniería de software experimentará una bifurcación de dos velocidades: una velocidad comercial asistida, orientada a la agilidad y la experiencia de usuario (Web/Mobile), y una velocidad de integridad manual, dedicada a la infraestructura crítica que sostiene la seguridad física humana. Esta dualidad define el estado del arte actual, donde la productividad no puede comprometer la previsibilidad de los sistemas de misión crítica.

Caso 2: El Modelo Asistido (Copilot / Autocomplete)

El segundo paradigma identificado se fundamenta en una dinámica de programación en pares (*Pair Programming*), donde el humano asume el rol de conductor estratégico y la Inteligencia Artificial opera como un copiloto táctico especializado en la generación de *boilerplate* y sintaxis rutinaria. No obstante, este esquema introduce riesgos cognitivos profundos. Investigaciones sobre el comportamiento del desarrollador advierten sobre el Sesgo de Automatización: una tendencia psicológica a validar sugerencias incorrectas o inseguras debido a la confianza excesiva en el sistema automatizado, lo que reduce la vigilancia crítica a medida que aumenta la fatiga cognitiva (Mozannar et al., 2024).

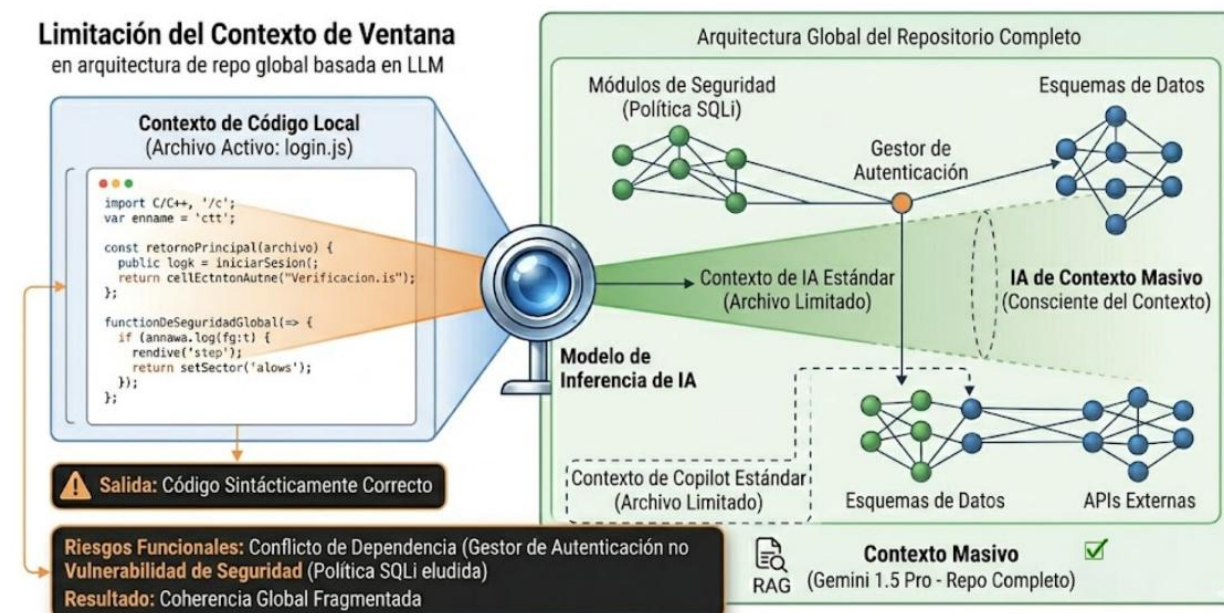
Desde una perspectiva técnica, el mecanismo de predicción probabilística de tokens mediante arquitecturas *Transformer* (como OpenAI Codex) presenta una vulnerabilidad estructural: la ventana de

contexto restringida. Las herramientas de asistencia estándar poseen una capacidad de procesamiento limitada que solo abarca el archivo activo o fragmentos de pestañas adyacentes, careciendo de un grafo de conocimiento semántico del repositorio completo. Como señalan Chen et al. (2021), el modelo optimiza la probabilidad local de la siguiente palabra en lugar de la corrección lógica global, resultando en alucinaciones sintácticamente perfectas, pero funcionalmente erróneas.

Figura 4

Limitación de la Ventana de Contexto y el Fenómeno de la Visión de Túnel

DIAGRAMA DEL CONTEXTO ARQUITECTÓNICO LOCAL VS. GLOBAL EN ASISTENTES DE PROGRAMACIÓN CON IA

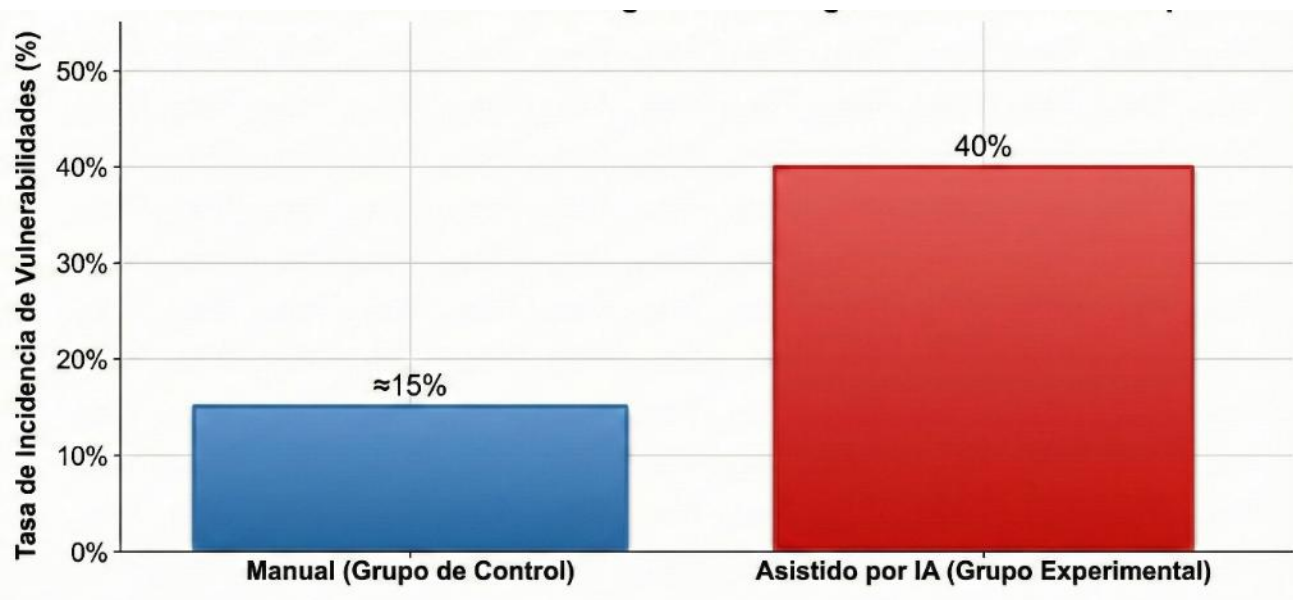


Elaboración propia. Análisis comparativo de la asimetría de inferencia entre asistentes de código de primera generación (izquierda) y modelos de orquestación sistémica (derecha). El diagrama evidencia cómo la restricción del contexto local induce a una Visión de Túnel y fragmentación de la coherencia, mientras que el procesamiento del grafo completo del repositorio (Caso 3) garantiza la integridad arquitectónica global.

Esta fragmentación de la coherencia arquitectónica no es solo un problema de eficiencia, sino que compromete la integridad del proyecto. Investigaciones de la Universidad de Stanford confirman que esta carencia de visión sistémica induce a los desarrolladores a introducir vulnerabilidades críticas, como inyecciones SQL o XSS, al omitir políticas de seguridad definidas fuera del contexto local. Esta disparidad empírica se cuantifica en la Figura 5, donde se observa un incremento alarmante en la tasa de incidentes en entornos asistidos.

Figura 5

Incidencia de Vulnerabilidades CWE: Desarrollo Manual vs. Asistido por IA



Elaboración propia basada en los datos experimentales de Pearce et al. (2022).

El análisis de la Deuda de Seguridad visualizada en la Figura 5 evidencia que la disparidad en la tasa de vulnerabilidades, un 40% en el grupo asistido frente a un 15% en el grupo manual, no es solo producto de sugerencias defectuosas, sino de la claudicación de la auditoría humana. El programador, seducido por la corrección sintáctica de la IA, omite la validación profunda de vectores de ataque complejos. Esto confirma la hipótesis central de esta sección: la velocidad ganada en la fase de generación se traduce en un interés compuesto de riesgo de ciberseguridad y deuda técnica que deberá ser subsanado con un costo significativamente mayor en fases posteriores del ciclo de vida de desarrollo de software (SDLC).

Caso 3: El Modelo de Ingeniería Generativa Sistémica

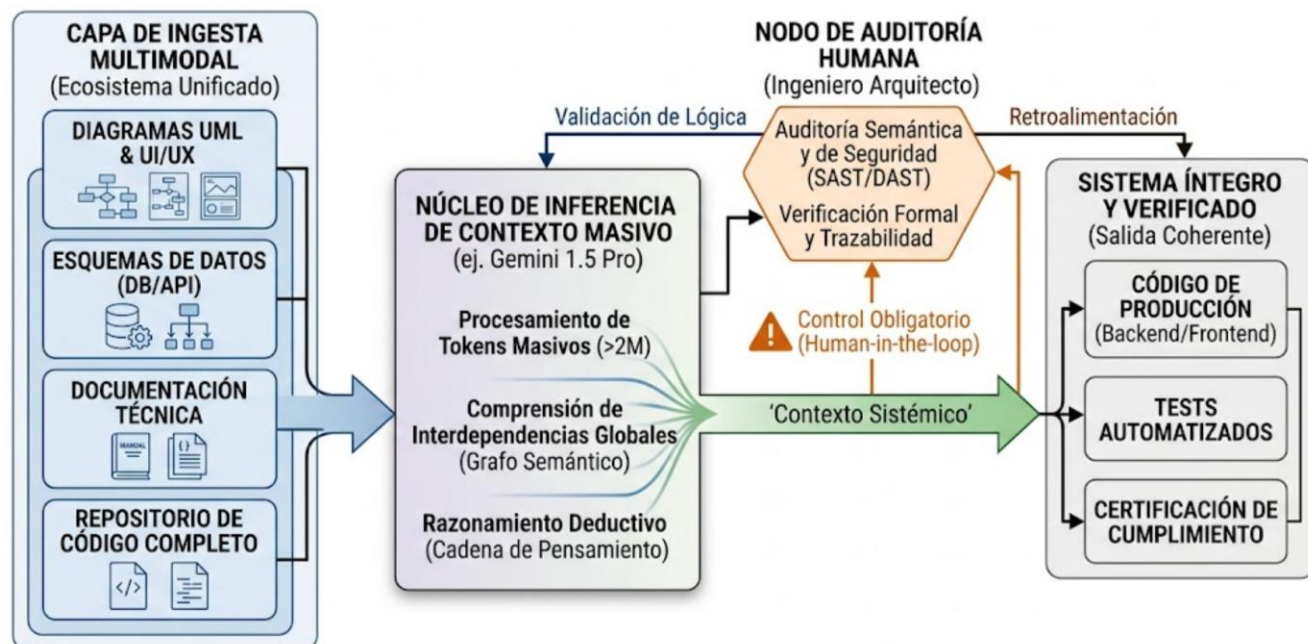
Como culminación del análisis, se propone una transición disruptiva del rol del programador tradicional hacia el de Arquitecto de Sistemas Orquestados por IA. En este paradigma, el ser humano trasciende la implementación sintáctica atomizada, la codificación mecánica de estructuras básicas, para concentrarse en la definición de intenciones, restricciones arquitectónicas y la validación de la lógica de negocio. Bajo este esquema, el ingeniero actúa como un auditor de alto nivel que interactúa con el repositorio completo, delegando la construcción de módulos complejos a agentes de IA con capacidad de razonamiento deductivo. Esta reorientación estratégica mitiga la carga cognitiva de la implementación para redirigirla hacia la Auditoría Semántica y de Seguridad (Dohmke et al., 2023).

La viabilidad técnica de este modelo converge en dos pilares fundamentales: las Ventanas de Contexto Masivo y el Razonamiento Multimodal. Modelos de última generación, como Gemini 1.5 Pro,

poseen la capacidad de procesar volúmenes de información superiores a los 2 millones de tokens, permitiendo la ingesta íntegra de proyectos de software, documentación y librerías en una sola operación de inferencia. Este avance tecnológico erradica la visión de túnel característica de los asistentes de primera generación, permitiendo que la IA comprenda las interdependencias globales y la semántica profunda del sistema.

Figura 6

Arquitectura de Inferencia de Contexto Masivo y Multimodal: El Modelo Sistémico



Elaboración propia basada en la arquitectura de modelos de contexto largo (Reid et al., 2024). La ilustración esquematiza el flujo del Modelo Generativo Sistémico, donde el núcleo de inferencia procesa simultáneamente múltiples modalidades de información bajo un bucle de control humano obligatorio, garantizando una generación alineada con la lógica de negocio global.

La integración de estas capacidades multimodales permite que la IA interprete diagramas estructurales (UML) e interfaces de usuario (UI/UX) como entradas directas para la generación de código, cerrando la brecha histórica entre el diseño arquitectónico y la implementación técnica (Reid et al., 2024). Este flujo de trabajo alcanza una optimización exponencial en la entrega de valor, facilitando que equipos reducidos, denominados en la literatura como *One-Person Unicorns*, gestionen sistemas de complejidad empresarial con una agilidad sin precedentes.

Sin embargo, este incremento en la capacidad de generación establece una nueva frontera de responsabilidad: la Verificación Formal. Dado que la IA genera código a velocidades sobrehumanas, el cuello de botella del desarrollo se desplaza ineludiblemente hacia la capacidad humana de auditar la seguridad y la alineación con los requisitos. El riesgo emergente es la proliferación de "Sistemas Caja Negra" cuya lógica interna resulte inescrutable para la propia organización si se descuida la supervisión

experta (Hou et al., 2024). Por lo tanto, el Modelo Sistémico no solo requiere tecnología avanzada, sino un nuevo contrato ético donde el ingeniero asegura la integridad verificable del sistema.

Para sintetizar esta evolución, la Tabla 2 contrasta las dimensiones operativas y cognitivas de los tres paradigmas analizados.

Tabla 2
Evolución de los Paradigmas de Desarrollo de Software: Dimensiones Operativas y Cognitivas.

Dimensión	Caso 1: Determinista (Manual)	Caso 2: Asistido (Copilot)	Caso 3: Generativo Sistémico (Propuesto)
Unidad de Trabajo	Línea de código / Función.	Fragmento de código (Snippet).	Módulo completo / Sistema.
Rol del Humano	Traductor (Requisito -> Sintaxis).	Editor / Revisor de sugerencias.	Orquestador y Auditor.
Contexto de la IA	Nulo (No existe).	Local (Archivo actual).	Global (Repositorio Completo + Docs).
Riesgo Principal	Fatiga humana / Error de sintaxis.	Incoherencia arquitectónica / Seguridad.	Opacidad ("Caja Negra") si no se audita.
Productividad	Lineal (1x).	Incremental (1.3x - 1.5x).	Exponencial (5x - 10x).

Elaboración propia. La tabla sintetiza el salto cualitativo hacia el modelo sistémico, destacando la transición de la Carga Cognitiva de Sintaxis (escritura mecánica) hacia la Carga Cognitiva de Auditoría (verificación estratégica) como el nuevo eje de la ingeniería de software profesional. Esta comparativa resalta que el éxito de la ingeniería en la era generativa no reside en la velocidad de escritura, la cual se ha vuelto un commodity, sino en la capacidad estratégica del ingeniero para orquestar contextos masivos y certificar la integridad de sistemas estocásticos. Bajo este esquema de responsabilidad humana irrenunciable, el profesional deja de ser un traductor de lógica a código para convertirse en el garante de la coherencia semántica y de seguridad del sistema completo.

Conclusiones

La presente investigación confirma de manera categórica la hipótesis planteada: la ingeniería de software atraviesa una ruptura epistemológica irreversible. La Inteligencia Artificial Generativa no emerge como un sustituto del factor humano, sino como el catalizador de una evolución ineludible del ingeniero, quien transita de un rol de constructor de sintaxis hacia el de un arquitecto y auditor de sistemas estocásticos. Se concluye que la disciplina ha fragmentado su naturaleza monolítica para bifurcarse en dos velocidades operativas: un Paradigma Generativo, que hegemoniza el desarrollo comercial mediante una eficiencia exponencial, y un Paradigma Determinista, que persiste como el estándar irreductible en sistemas de misión crítica. Esta divergencia responde a una incompatibilidad ontológica fundamental entre la naturaleza probabilística de los LLM y los requisitos de trazabilidad absoluta exigidos por normativas de seguridad funcional como ISO 26262; en consecuencia, la codificación manual se mantiene como una



salvaguarda técnica en sectores donde la integridad física humana es el eje central, mientras persista la opacidad algorítmica de la Caja Negra.

Consecuentemente, el análisis del desarrollo asistido revela una paradoja de productividad de implicaciones sistémicas para las organizaciones. Si bien la integración de asistentes de codificación genera una aceleración inmediata en la fase de construcción, esta ventaja se ve neutralizada por una degradación erosiva de la calidad del software derivada del Sesgo de Automatización. Los hallazgos demuestran que la transferencia de confianza hacia sugerencias sintácticas, exacerbada por la visión de túnel de los asistentes estándar, incrementa de forma crítica la superficie de ataque y la introducción de vulnerabilidades. Bajo esta premisa, la eficiencia ganada en la implementación se transmuta, sin protocolos de gobernanza robustos, en una deuda técnica y un riesgo de ciberseguridad que se acumula bajo un esquema de interés compuesto durante las fases de mantenimiento y operación.

Frente a esta crisis de fiabilidad, la investigación identifica que la sostenibilidad del ecosistema de desarrollo futuro depende de la transición estratégica hacia un Modelo de Ingeniería Generativa Sistémica. La superación de la ceguera arquitectónica, lograda mediante la implementación de Ventanas de Contexto Masivo y razonamiento multimodal, constituye el requisito técnico *sine qua non* para la viabilidad de la IA en entornos de alta complejidad. Este avance reconfigura el valor de mercado del perfil profesional: la competencia del ingeniero deja de residir en la retentiva sintáctica para centrarse en la capacidad de orquestación sistémica y en la facultad crítica para auditar la lógica sintética, garantizando que el artefacto resultante se alinee con los objetivos estratégicos y éticos del negocio.

Finalmente, se establece que es imperativo renegociar el contrato social de la ingeniería de software, formalizando el Bucle Humano (*Human-in-the-loop*) como una barrera de calidad obligatoria, y no contingente, dentro del ciclo de vida del desarrollo (SDLC). Tanto la formación académica como la praxis profesional deben desplazar el énfasis tradicional de la escritura desde cero hacia competencias de nivel superior como la ingeniería de *prompts*, la verificación formal y la seguridad ofensiva. En una era de abundancia de código y degradación de los costes de producción, el activo más escaso y, por ende, el de mayor valor estratégico, será la integridad semántica y la confianza verificable de los sistemas inteligentes.

La transición hacia el nuevo paradigma de ingeniería demanda una respuesta coordinada entre la industria y la academia, fundamentada en cuatro ejes críticos. Primero, es imperativo establecer una gobernanza diferenciada mediante un Semáforo Tecnológico que restrinja el uso de IA estocástica en sistemas de misión crítica, preservando el determinismo manual donde la seguridad humana es prioritaria. Segundo, la educación superior debe evolucionar hacia la orquestación de sistemas, priorizando la auditoría y la arquitectura sobre la memorización sintáctica para formar ingenieros capaces de certificar integridad algorítmica.

Para garantizar la sostenibilidad operativa, se propone la auditoría obligatoria de caja blanca, integrando el bucle humano (*Human-in-the-loop*) con herramientas SAST para asegurar que la velocidad de generación jamás comprometa la capacidad de verificación. Finalmente, se insta a la adopción de arquitecturas de contexto masivo que erradiquen la ceguera arquitectónica, garantizando que cada intervención de la IA respete la coherencia global del proyecto. Esta hoja de ruta asegura que la eficiencia generativa se traduzca en progreso tecnológico y no en una deuda técnica inmanejable.

Referencias

- CISA. (2023). *Shifting the balance of cybersecurity risk: Principles and priorities for security-by-design and default*. Cybersecurity & Infrastructure Security Agency. <https://www.cisa.gov/resources-tools/resources/secure-by-design>
- Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., ... & Zaremba, W. (2021). *Evaluating large language models trained on code*. arXiv preprint arXiv:2107.03374. <https://doi.org/10.48550/arXiv.2107.03374>
- Dohmke, T., Dellangnol, I., & Pulley, C. (2023). *Sea change: The future of AI and software development*. GitHub. <https://github.blog/2023-06-27-the-economic-impact-of-the-ai-powered-developer-lifecycle-and-lessons-from-github-copilot/>
- GitHub. (2024). *The state of the octoverse 2024: AI and the evolution of the developer workflow*. GitHub News. <https://github.blog/news-insights/octoverse/octoverse-2024/>
- GitHub. (2024). *The state of the Octoverse: 2024 report on open source software*. GitHub Resources. <https://github.blog/octoverse/>
- GitClear. (2024). *Coding on Copilot: 2023 data suggests downward pressure on code quality*. <https://www.gitclear.com/coding-on-copilot-data-shows-ais-downward-pressure-on-code-quality>
- Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., ... & Lo, D. (2024). *Large language models for software engineering: A systematic literature review*. arXiv preprint arXiv:2308.10620. <https://doi.org/10.48550/arXiv.2308.10620>
- IBM Security. (2025). *X-Force Threat Intelligence Index 2025: The impact of generative AI on cyber resilience*. IBM Corporation. <https://www.ibm.com/reports/threat-intelligence>
- Knight, J. C. (2002). Safety-critical systems: Challenges and directions. *Proceedings of the 24th International Conference on Software Engineering (ICSE)*, 547-550. <https://doi.org/10.1145/581339.581406>
- Lakhani, K. R., & Sull, D. (2024). *The era of the AI-augmented engineer: Beyond syntactic coding*. Harvard Business Review. <https://hbr.org/2024/02/how-ai-is-changing-the-way-developers-work>
- McKinsey & Company. (2023). *Unleashing developer productivity with generative AI*. McKinsey Digital. <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai>
- Mozannar, H., Bansal, G., Fourney, A., & Horvitz, E. (2024). *Reading between the lines: Modeling user behavior and costs in AI-assisted programming*. arXiv preprint arXiv:2210.14306.
- Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2022). Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions. *2022 IEEE Symposium on Security and Privacy (SP)*, 754-768. <https://arxiv.org/abs/2108.09293>
- Peng, S., Kalliamvakou, E., Cihon, P., & Demarrer, M. (2023). *The impact of AI on developer productivity: Evidence from GitHub Copilot*. arXiv preprint arXiv:2302.06590.
- Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2023). Do users write more insecure code with AI assistants? *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2785-2799. <https://doi.org/10.1145/3548606.3560605>

- Reid, M., Savinov, N., Teplyashin, D., & Co-authors (Google DeepMind). (2024). *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. Google DeepMind Technical Report.
- Russo, D., Laredo, J. L., Storey, M. A., Hoda, R., Serebrenik, A., & Pandita, R. (2024). Generative AI in software engineering must be human-centered: The Copenhagen Manifesto. *Journal of Systems and Software*, 216, 112115. <https://dl.acm.org/doi/10.1016/j.jss.2024.112115>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is all you need*. Advances in Neural Information Processing Systems, 30.

Declaración de Autoría

Conceptualización: Juan Rigoberto Castillo Serracín y Javier Miguel Gómez S.; **Investigación:** Juan Rigoberto Castillo Serracín y Javier Miguel Gómez S.; **Metodología:** Juan Rigoberto Castillo Serracín y Javier Miguel Gómez S.; **Redacción – Borrador orogianal:** Juan Rigoberto Castillo Serracín y Javier Miguel Gómez S.; **Redacción – revisión y edición:** Juan Rigoberto Castillo Serracín y Javier Miguel Gómez S.; **Autor de correspondencia:** Javier Miguel Gómez S.

Conflicto de interés

Los autores de este manuscrito certifican que no existen conflictos de interés de ningún tipo, ni financieros ni personales, académicos o políticos, que pudieran haber influido de manera inapropiada en la realización, análisis o interpretación de los resultados de esta investigación.

Información adicional

La correspondencia y las solicitudes de materiales sobre este escrito deben dirigirse al autor de correspondencia al correo electrónico proporcionado.

Las impresiones y la información sobre permisos están disponibles en el siguiente enlace: https://www.revistas.up.ac.pa/index.php/contacto/acceso_reuso