CODEX HASH

RESEARCH LABORATORY

# Una frase antes del plan

*Un estudio confirmatorio prerregistrado del pre-mortem como prompt en agentes de código con LLM*

**Carlos Ulisses Flores**

Codex Hash Research Laboratory

ORCID: 0000-0002-6034-7765

Codex Hash Research Report 2026-03

Fecha: 17 de noviembre de 2026

**Resumen**

Un pre-mortem pide al equipo que suponga que el plan fracasó y enumere los motivos (Klein, 2007). En un estudio prerregistrado anterior, la mitad de enumeración de ese ejercicio, puesta en una frase, hizo que el plan de un agente de código cubriera más riesgos enumerados por separado, como resultado secundario (Flores, 2026a). Este es el estudio confirmatorio. Claude Code, en modo de planificación en Claude Opus 5.5 y Claude Sonnet 5.5, recibió doce solicitudes de cambio en repositorios públicos fijados, bajo dos solicitudes que difieren en una frase: “Plan this change.” y “Before planning, list the ways this change could go wrong.” Tres modelos de otros proveedores escribieron a ciegas los riesgos de referencia, y un juez ciego de otro proveedor puntuó cada respuesta. Protocolo, listas y análisis se sellaron, con sello temporal, antes de la primera ejecución. En 240 ejecuciones, la frase elevó la cobertura del 46,1% al 57,5%, diferencia media de 11,4 puntos porcentuales en 24 estratos de tarea por modelo (permutación estratificada, $p < ,001$, también en cada modelo). Un segundo juez dio +10,7 puntos ($\kappa$ de Cohen = ,86). Sin los tres estratos con ejecución perdida por error de red, la diferencia es de 10,1 puntos. Frente a un segundo conjunto de listas, pedido con otras palabras, es de 6,7 puntos ($p < ,001$), y 9,0 según un juez posterior de un tercer proveedor. La afirmación se limita a dos modelos de un proveedor, a listas escritas por modelos y a planes, no a código.

**Palabras clave:** pre-mortem; ingeniería de *prompts*; agentes de código basados en LLM; identificación de riesgos; prerregistro; planificación de software

**Abstract**

A pre-mortem asks a team to assume that its plan has failed and to list why (Klein, 2007). In an earlier pre-registered study the listing half of that exercise, put in one sentence, made a coding agent's plan cover more independently listed risks, as a secondary result (Flores, 2026a). This paper reports the confirmatory study. Twelve change requests on pinned public repositories were given to Claude Code in plan mode, on Claude Opus 5.5 and Claude Sonnet 5.5, under two requests that differ by one sentence: "Plan this change." and "Before planning, list the ways this change could go wrong." Reference risks were written blind by three models of other vendors, and a blind judge of another vendor scored each answer. Protocol, lists and analysis were sealed and timestamped before the first trial. Over 240 trials the sentence raised coverage from 46.1% to 57.5%, a mean difference of 11.4 percentage points over 24 task-by-model strata (stratified permutation $p < .001$, also within each model). A second judge gave +10.7 points (Cohen's $\kappa = .86$). Without the three strata that hold a trial lost to a network error the difference is 10.1 points. Against a second set of reference lists, worded differently, it is 6.7 points ($p < .001$), and 9.0 by a third-vendor judge added later. The claim is limited to two models of one vendor, to model-written reference lists and to plans, not code.

**Keywords:** pre-mortem; prompt engineering; LLM coding agents; risk identification; pre-registration; software planning

## Índice

1. Introducción ..... 5
1.1. Contribuciones ..... 8
2. Fundamentación y trabajos relacionados ..... 9
2.1. El pre-mortem en personas ..... 9
2.2. Enumerar el fallo antes de responder, en LLM y agentes ..... 10
2.3. Recordatorios y predicciones en la generación de código ..... 11
2.4. Instrucciones cortas y sensibilidad al *prompt* ..... 12
2.5. Dónde se sitúa este estudio ..... 12
3. El estudio de origen ..... 13
4. Método ..... 15
4.1. Hipótesis y diseño ..... 15
4.2. Tareas ..... 15
4.3. Condiciones y *harness* ..... 16
4.4. Riesgos de referencia, escritos a ciegas ..... 20
4.5. Puntuación ..... 21
4.6. Análisis y regla de decisión ..... 23
4.7. Prerregistro y registro de integridad ..... 24
5. Resultados ..... 28
5.1. Recolección ..... 28
5.2. Resultado principal ..... 28
5.3. Segundo juez ..... 31
5.4. Relevancia ..... 31
5.5. Sensibilidad a las tres ejecuciones con error ..... 32
5.6. Costo ..... 34
5.7. Qué riesgos añade la frase (exploratorio) ..... 35
5.8. Una segunda vara de medir, con otras palabras (exploratorio) ..... 36
6. Discusión ..... 40
7. Objeciones y amenazas a la validez ..... 42
8. Desviaciones del protocolo ..... 48
9. Conclusión ..... 50
10. Disponibilidad de datos y código ..... 51
11. Declaraciones ..... 52
Referencias ..... 53
Apéndice A — Las solicitudes exactas ..... 58
Apéndice B — Una lista de referencia completa ..... 59
Apéndice C — Los textos en inglés, traducidos ..... 60
Nota Técnica ..... 62
Sobre el autor ..... 63

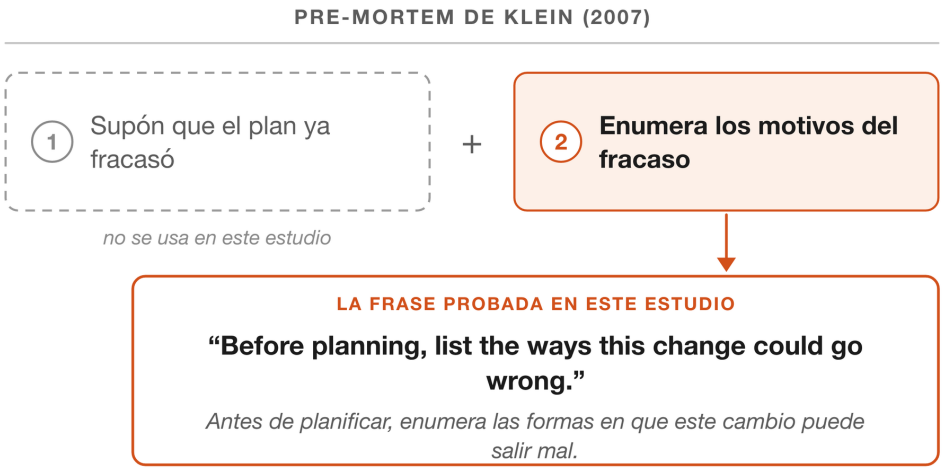
## 1. Introducción

Los agentes de código basados en grandes modelos de lenguaje (LLM) ya planifican y ejecutan cambios en repositorios reales (Jimenez et al., 2024). Los planes estudiados aquí se piden antes de que se escriba código alguno, un punto temprano para detectar un riesgo olvidado. Este artículo pone a prueba una afirmación: abrir una solicitud de planificación con la frase "Before planning, list the ways this change could go wrong." (en español: "Antes de planificar, enumera las formas en que este cambio puede salir mal."), en lugar de "Plan this change.", hace que la tabla de riesgos que cierra el plan del agente cubra más de los riesgos que esperan revisores independientes. La afirmación es estrecha a propósito, porque los efectos de la redacción de un *prompt* a menudo no se mantienen de un modelo o conjunto de tareas al siguiente (Sclar et al., 2024).

La frase viene de la investigación sobre decisión. En un *pre-mortem*, el equipo imagina que el plan ya fracasó y anota los motivos (Klein, 2007). La técnica se apoya en la *retrospectiva prospectiva*: quien explica un desenlace como si ya hubiera ocurrido genera más motivos que quien explica un desenlace que puede ocurrir (Mitchell et al., 1989). Las normas de evaluación de riesgos enumeran técnicas estructuradas vecinas, como el análisis de modos de fallo y sus efectos, entre los métodos que una organización puede aplicar antes de actuar (International Electrotechnical Commission, 2019). La frase probada aquí conserva solo la segunda mitad del ejercicio de Klein, la enumeración, y abandona la suposición de fracaso (Figura 1).

**Figura 1**

*Las dos mitades de un pre-mortem y la que conserva la frase probada*




Para los LLM, la evidencia más cercana está en la seguridad, donde hacer que el modelo enumere los daños posibles antes de responder reduce la nocividad juzgada más que ordenarle ser seguro (Kim et al., 2026). En agentes, anticipar el fallo antes de actuar se incorporó a métodos de varios pasos y se evaluó como un todo (Wang et al., 2024; Wang et al., 2026). En la generación de código, los recordatorios para ser seguro dan resultados mixtos o nulos (Tony et al., 2025; Kharma et al., 2026), y un *prompt* más largo que pide a un agente de código que prevea casos límite ayudó a dos modelos y perjudicó a un tercero (Patel et al., 2026). Falta una medición controlada de la dosis mínima en la planificación de software: una frase, frente a la misma solicitud sin ella, puntuada contra riesgos que otra persona enumeró.

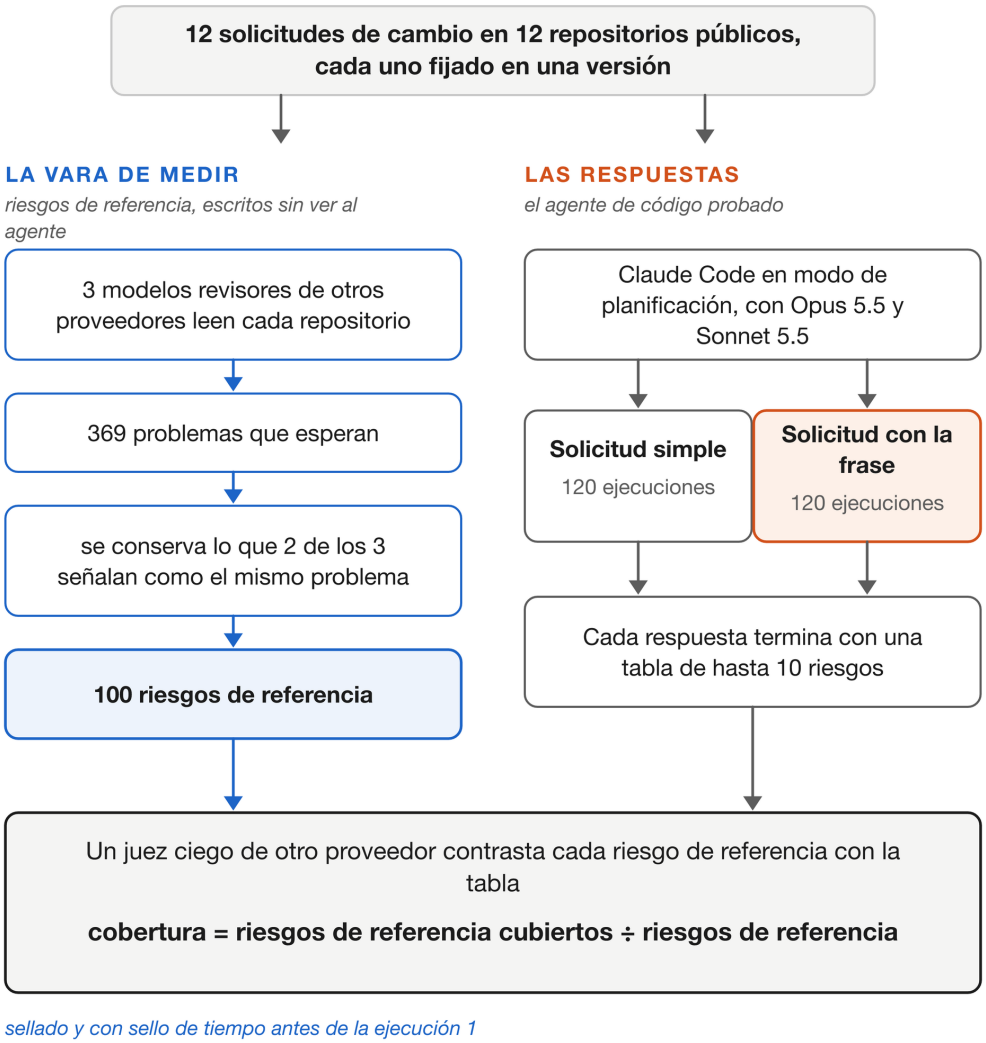
La frase surgió de una evaluación anterior de una herramienta que mantiene el autor. ASP es un *plugin* de código abierto, un complemento instalable, para Claude Code, el agente de programación de línea de comandos de Anthropic; guía al agente por un protocolo de pre-mortem de 13 fases, y su medición prerregistrada inicial, aquella en que el protocolo y el análisis se fijaron en público antes de recoger los datos (Sección 4.7), no halló beneficio medible: el protocolo completo cubrió menos riesgos de referencia que una solicitud simple con la frase añadida (Flores, 2026a). La frase,

a su vez, superó a la solicitud simple en 10 puntos porcentuales. Esa comparación se prerregistró como secundaria, en un solo modelo y en seis tareas, de modo que justificaba un nuevo estudio y nada más (Nosek et al., 2018).

Este artículo presenta ese estudio. Se diseñó para dar al hallazgo anterior una oportunidad justa de fallar: doce tareas nuevas, un segundo modelo, la comparación como única pregunta principal y una regla de decisión sellada antes de que existiera cualquier dato (Figura 2).

**Figura 2**

*Diseño del estudio: cómo se producen y se puntúan los riesgos de referencia y las respuestas*



### 1.1. Contribuciones

1. **Una prueba confirmatoria y prerregistrada de una solicitud de una frase en un agente de código.** En 240 ejecuciones, en doce repositorios públicos fijados y dos modelos, la frase eleva la cobertura de los riesgos enumerados de forma independiente en 11,4 puntos porcentuales ($p < ,001$), y en un valor semejante dentro de cada modelo (Sección 5.2).

2. **Comprobaciones de la mejora.** Un segundo juez de otro proveedor llega a la misma decisión (Sección 5.3), los riesgos enumerados no se juzgan menos concretos (Sección 5.4), y el resultado sobrevive a la retirada de tres ejecuciones que un error de red puso en cero a favor de la comparación (Sección 5.5). Frente a listas de referencia pedidas después con otras palabras, la ganancia sigue siendo positiva y baja a 6,7 puntos (Sección 5.8).

3. **Un registro sellado y abierto.** Cada *prompt* definido por el estudio, cada lista de referencia, respuesta, veredicto y archivo de análisis es público y está ligado a un registro encadenado por hash (*ledger*), una cadena de huellas digitales de archivos, cuyo estado anterior a las ejecuciones está anclado en la *blockchain* de Bitcoin, un registro público que es impracticable reescribir (Sección 4.7), y un único comando recalcula cada resultado del estudio (Flores, 2026b).

4. **Una lista declarada de objeciones.** La Sección 7 toma las objeciones más fuertes al resultado, incluidas las planteadas por revisores externos de este manuscrito, y responde a cada una con datos o la registra como abierta.

Hasta donde sabemos, ningún estudio anterior mide el efecto de una instrucción mínima de pre-mortem sobre la cobertura de riesgos planteados de forma independiente en la planificación de cambios de software por un agente de código, aparte del estudio de origen, donde fue una comparación secundaria (Flores, 2026a). La Sección 2.5 dice qué delimita esa afirmación. El mecanismo en sí, enumerar modos de fallo antes de responder, tiene respaldo anterior (Kim et al., 2026) y no se reivindica aquí.

## 2. Fundamentación y trabajos relacionados

### 2.1. El pre-mortem en personas

Klein (2007) describe el pre-mortem de proyecto como una reunión breve antes de que empiece el trabajo: se supone que el plan fracasó, y cada participante escribe los motivos. El experimento de apoyo es el de Mitchell et al. (1989), en el que la certeza sobre un desenlace, más que su posición en el tiempo, aumentó el número de explicaciones que las personas produjeron. En un experimento controlado con 178 estudiantes, el pre-mortem redujo la confianza en el plan en 25,0 puntos, frente a 14,0 de la lista de pros y contras y 12,4 de la lista solo de contras (Veinott et al., 2010). Los efectos humanos establecidos están, por tanto, en la cantidad y en la confianza, de modo que un estudio con LLM necesita comprobar que más elementos no signifiquen simplemente elementos más vagos (Sección 5.4).

El procedimiento humano tiene dos partes, la suposición de fracaso y la enumeración de los motivos (Klein, 2007). Un *prompt* puede pedir la segunda sin la primera, y la frase estudiada aquí hace exactamente eso. La comparación entre respuestas humanas y de IA en la planificación por pre-mortem empezó en la investigación de factores humanos (Veinott y Lehman, 2024), pero como comparación de quién responde, no de cómo se instruye al modelo.

### 2.2. Enumerar el fallo antes de responder, en LLM y agentes

InvThink es el trabajo anterior más cercano (Kim et al., 2026). El modelo enumera los daños posibles, analiza sus consecuencias y solo entonces responde bajo las restricciones que derivó. En el modo solo de *prompt*, sin entrenamiento, el método reduce una nota de nocividad juzgada de 3,22 a 2,19, mientras que una instrucción de seguridad de tamaño parecido llega solo a 2,62. La explicación de los autores es que la enumeración explícita saca a la luz conocimiento que una solicitud directa deja sin uso. El desenlace allí es contenido nocivo, y el marco tiene tres pasos.

En agentes, Devil's Advocate hace que un agente de navegación web reflexione sobre probables fallos y remedios alternativos antes de cada acción, y reporta un 23,5% de éxito en las tareas frente al 19,8% de una línea de base de planificar y actuar, con un 45% menos de revisiones de plan que el mejor método competidor (Wang et al., 2024). PreFlect critica el plan antes de la ejecución usando errores de planificación destilados de trayectorias anteriores, y eleva el éxito en un *benchmark* de asistente general, un conjunto fijo de tareas de prueba, de 46,1% a 58,2% (Wang et al., 2026). Ambos evalúan métodos de varios pasos; ninguno prueba una instrucción de una frase, y ninguno cuenta cuántos riesgos esperados nombra un plan.

Una tercera línea trata de lo que un modelo sin instrucción deja fuera. En un experimento de campo pequeño y preliminar sobre planificación de *sprint*, en la que un equipo planifica su próximo ciclo corto de trabajo, un LLM que planificaba solo documentó el 36,4% de los riesgos que después se materializaron, 4 riesgos, frente al 78,6% de los planificadores humanos, 11 riesgos (Caraeni et al., 2026). Los agentes LLM también pueden producir muchas más consecuencias candidatas de un producto que los especialistas trabajando solos (Fröhling et al., 2026). El resultado de campo es la brecha que un *prompt* de planificación podría estrechar, y su desenlace es vecino del que se

usa aquí: allí, captura de riesgos que se materializaron; aquí, captura de riesgos que esperaban los revisores.

### 2.3. Recordatorios y predicciones en la generación de código

Varios estudios añaden una instrucción de cautela a un *prompt* de generación de código y miden el código. En GPT-4, algunas técnicas de *prompt* redujeron la densidad de debilidades de seguridad entre un 59% y un 78% (Tony et al., 2025). En cinco modelos y cuatro métodos de *prompt*, otro estudio no halló una reducción estadísticamente significativa de vulnerabilidades (Kharma et al., 2026). Un *prompt* de sistema que pedía cobertura de casos límite a menudo no fue mejor que la línea de base en la generación de código sin ejemplos (Cheng y Mastropaolo, 2026). Estos resultados advierten contra suponer que una frase corta de cautela ayude.

También difieren del escenario actual de un modo que importa. Sus instrucciones más simples son recordatorios ("sé seguro"), aunque las técnicas comparadas incluyen también las iterativas, como la crítica y la mejora recursivas (Tony et al., 2025), y el desenlace es el código. Una instrucción de *enumerar* es un acto distinto de una instrucción de *tener cuidado*, distinción que InvThink traza frente a su línea de base de *prompt* de seguridad (Kim et al., 2026). Pedir al modelo que enumere sus supuestos implícitos es un acto de enumeración vecino, estudiado para la generación de código con un *prompt* estructurado (Wu, 2026).

El estudio más cercano en software es el de Patel et al. (2026). Dan a agentes de código, entre ellos Claude Code, un *prompt* de varias frases que pide al agente que prevea los casos límite que su implementación podría pasar por alto, antes de empezar, durante la implementación y antes de entregar. El desenlace está en el código: una tasa conjunta de parches (cambios de código propuestos) que resuelven la tarea y además evitan un defecto que un cambio posterior del proyecto

tuvo que corregir. En su comparación, esa tasa subió de 24,5% a 32,6% en Claude Opus 5, subió en un segundo modelo y cayó de 17,0% a 13,7% en Claude Sonnet 5, mientras que la parte de tareas resueltas cayó en los tres. Su resultado es la razón por la que la afirmación actual no se extiende de planes a código, y muestra que el efecto de esos *prompts* puede depender del modelo.

### 2.4. Instrucciones cortas y sensibilidad al *prompt*

Que una frase pueda mover la salida de un modelo está bien establecido. “Let’s think step by step” elevó la exactitud sin ejemplos en un *benchmark* aritmético de 17,7% a 78,7% (Kojima et al., 2022), y el *prompt* Plan-and-Solve refinó la misma idea al pedir que el modelo elabore antes un plan (Wang et al., 2023). La misma literatura muestra por qué tales hallazgos necesitan replicación: la exactitud varió hasta 76 puntos entre elecciones de formato sin significado alguno (Sclar et al., 2024).

Tampoco un *prompt* de pre-mortem ayuda en todas partes. En una comparación prerregistrada de 38 *prompts* para preguntas de pronóstico, que incluía un *prompt* de pre-mortem, la mayoría de los *prompts* cambió poco la exactitud (Schoenegger et al., 2025). Un efecto de redacción visto una vez, en un modelo y en un puñado de tareas, puede ser ruido de esa sensibilidad. Esa es la razón metodológica de un estudio confirmatorio con un segundo modelo, tareas nuevas y una regla fijada de antemano (Nosek et al., 2018).

### 2.5. Dónde se sitúa este estudio

La combinación estudiada aquí es: un escenario de ingeniería de software con repositorios reales; un desenlace definido como cobertura de una lista de referencia que el agente probado nunca vio; una dosis de una frase; y un resultado anterior en el que un protocolo largo y estructurado cubrió el 48% de los riesgos de referencia frente al 66% de esa frase (Flores, 2026a). Los vecinos más

cercanos difieren en al menos uno de esos puntos: InvThink en el dominio y en el desenlace (Kim et al., 2026), y Patel et al. (2026) en la dosis y en el desenlace.

La búsqueda bibliográfica detrás de la declaración de novedad de la Sección 1.1 abarcó revistas y servidores de *preprints* en inglés hasta el 14 de noviembre de 2026, con consultas adicionales en portugués, alemán, japonés y chino; su registro, con la situación de cada fuente, está en el repositorio (Flores, 2026b). Trabajó sobre títulos y resúmenes, de modo que un estudio que pruebe tal condición sin nombrarla allí, o uno que no esté indexado, puede haberse escapado.

## 3. El estudio de origen

El estudio anterior hizo 90 ejecuciones en seis repositorios públicos con Claude Opus 5.5, bajo tres condiciones: la solicitud simple, la solicitud con la frase y el protocolo ASP (Flores, 2026a, Sección 5.5). La Tabla 1 y la Figura 3 dan el resultado.

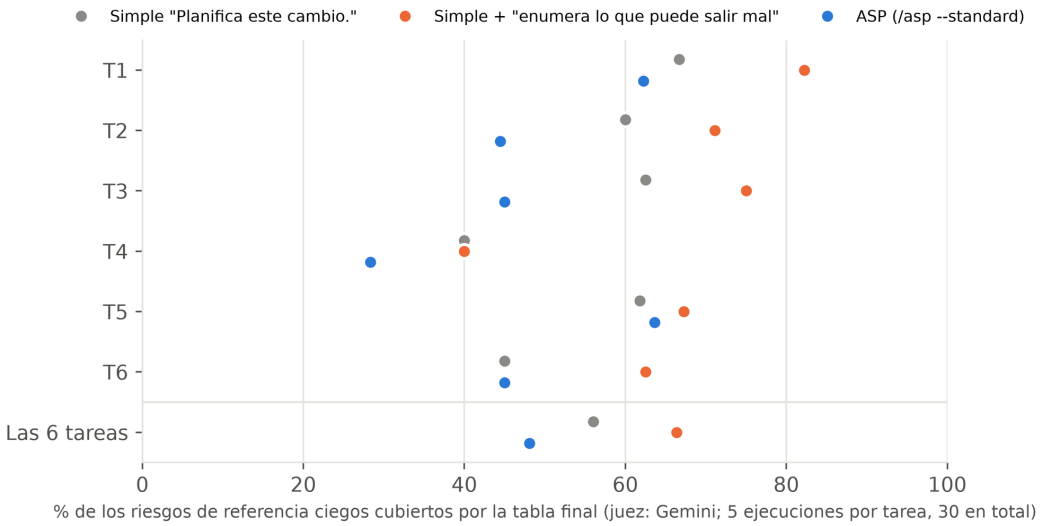
**Tabla 1**
*Cobertura de riesgos y tiempo por ejecución en el estudio de origen (2026-11), por condición*

| Condición | Cubiertos (juez principal) | Cubiertos (segundo juez) | Tiempo medio por ejecución |
|---|---|---|---|
| Solicitud simple | 56% | 57% | 58 s |
| Solicitud simple más la frase | 66% | 70% | 50 s |
| Protocolo ASP (13 fases) | 48% | 54% | 175 s |

*Nota.* 30 ejecuciones por condición en seis tareas. Fuente: Flores (2026a).

**Figura 3**

*Estudio de origen: fracción de los riesgos de referencia ciegos cubierta por la tabla final de cada condición, por tarea y en total (juez principal)*




*Nota.* En la leyenda, "/asp --standard" ejecuta el protocolo ASP y "enumera lo que puede salir mal" representa la frase.

La comparación principal prerregistrada de ese estudio, ASP contra la frase, fue negativa en las seis tareas: −18 puntos, p = 1,0 (Figura 3). La comparación secundaria, la frase contra la solicitud simple, fue de +10 puntos (p = ,001) según el juez principal y de +13 puntos (p < ,001) según el segundo (Tabla 1). Como el protocolo etiquetó esa comparación como secundaria, el informe anterior afirmó solo su dirección y anunció el presente estudio (Flores, 2026a).

Dos propiedades de ese resultado moldearon el nuevo diseño. El efecto se midió en un solo modelo, y los efectos de *prompt* pueden invertirse entre modelos (Patel et al., 2026). Y se apoyaba en seis tareas, de modo que el nuevo estudio usa doce que el anterior nunca tocó (Tabla 2).

# 4. Método

## 4.1. Hipótesis y diseño

La hipótesis es direccional: la frase eleva la cobertura. El diseño cruza doce tareas, dos modelos y dos condiciones, con cinco réplicas por celda, para un total de 240 ejecuciones (Figura 2). Un *estrato* es una tarea en un modelo; son 24, cada uno con cinco ejecuciones por condición. El protocolo fijó cada elemento de esta sección antes de la primera ejecución, siguiendo la práctica de separar el análisis confirmatorio del exploratorio (Nosek et al., 2018), y su archivo está en el repositorio (Flores, 2026b).

## 4.2. Tareas

Cada tarea es una solicitud de cambio de una frase en un repositorio público fijado en un *commit*, una versión registrada de su código (Tabla 2). Las solicitudes se escribieron para tener consecuencias más allá de los archivos que sugiere una primera lectura: datos guardados que deben migrar, clientes que deben seguir funcionando, valores por defecto de los que dependen otros proyectos. Abarcan siete lenguajes, una variedad que las directrices para estudios con LLM en ingeniería de software recomiendan en lugar de una muestra de un solo lenguaje (Baltes et al., 2027). El archivo de tareas se registró en un *commit* antes de que se escribiera cualquier lista de referencia (Sección 4.7).

### 4.3. Condiciones y *harness*

Las dos condiciones envían el mismo texto, salvo por una frase (el Apéndice A da las dos completas):

- **Simple:** la solicitud de cambio, luego "Plan this change." (en español: "Planifica este cambio.")
- **Frase:** la solicitud de cambio, luego "Before planning, list the ways this change could go wrong." (en español: "Antes de planificar, enumera las formas en que este cambio puede salir mal."; Tabla A1; Figura 4)

**Figura 4**
*Las dos solicitudes, lado a lado: una frase difiere y todo lo demás es idéntico*

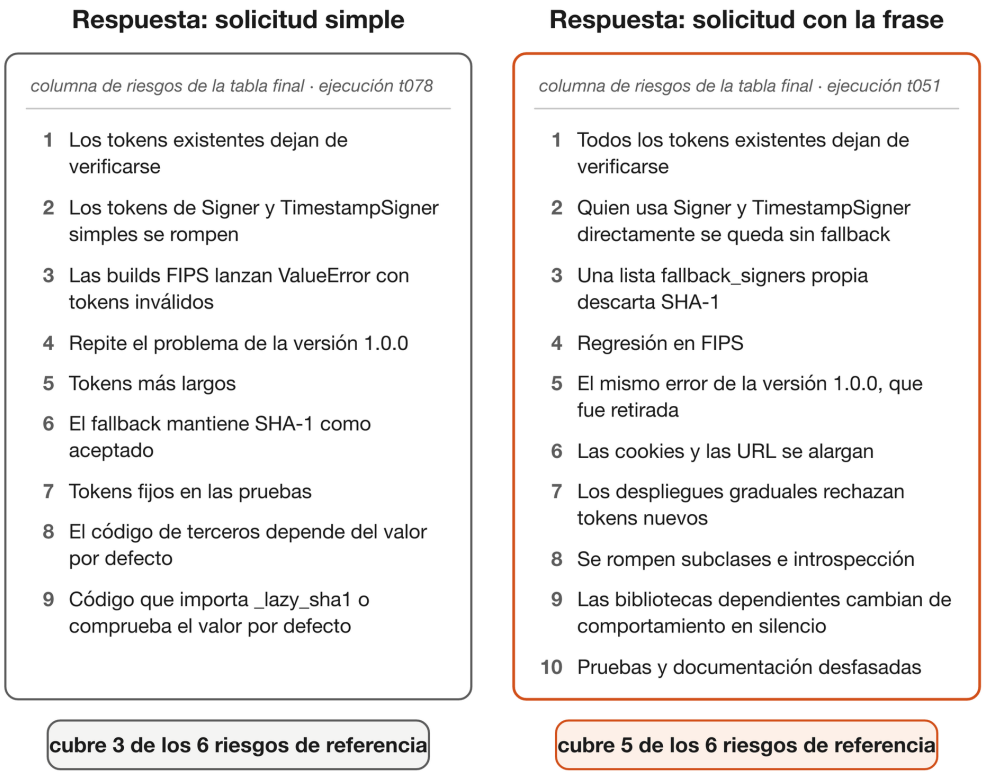



Ambas llevan a continuación el mismo *contrato de salida*, un párrafo fijo que ordena al agente declarar supuestos en lugar de hacer preguntas, no editar ningún archivo del proyecto (el modo de planificación aún puede guardar el propio plan en un archivo; Sección 5.6), responder en inglés y

terminar con los principales riesgos del cambio en una tabla de como máximo diez filas (Apéndice A). El contrato importa para leer el resultado. La condición simple no es un agente al que nunca se le preguntó por el riesgo: las dos condiciones deben terminar con una tabla de riesgos. La comparación mide, por tanto, el efecto conjunto de abrir la solicitud con la frase en lugar de la instrucción simple, además de la obligación de reportar riesgos al final; no separa la posición de la frase de su redacción (Sección 7, objeción 5). Los *prompts* son, byte a byte, los del estudio de origen (Flores, 2026a). La Figura 5 muestra lo que vuelve: la columna de riesgos de dos respuestas reales a la misma solicitud, una por condición.

**Figura 5**

*La columna de riesgos de dos respuestas reales a la solicitud de la Figura 4, una por condición (tarea U08, Claude Opus 5.5)*




*traducido del inglés; el original está en el repositorio*

En la Figura 5, FIPS son las normas federales de procesamiento de información de Estados Unidos (Federal Information Processing Standards); un sistema construido en modo FIPS rechaza los algoritmos criptográficos que esas normas no aprueban.

**Tabla 2**
*Las doce tareas*

| Tarea | Repositorio (lenguaje) | Solicitud de cambio |
|---|---|---|
| U01 | remix-run/blues-stack (TypeScript) | "Require new users to verify their email address before they can log in." |
| U02 | golang-gin-realworld-example-app (Go) | "Let authors change an article's title, and its URL slug with it, without breaking links that were already shared." |
| U03 | laravel-realworld-example-app (PHP) | "Let moderators hide abusive comments, and let the comment's author appeal within 7 days." |
| U04 | django-axes (Python) | "Lock users out by username and IP address together instead of by IP address alone, as the new default." |
| U05 | expressjs/session (JavaScript) | "Make sessions roll by default, so that every request extends the session's expiry." |
| U06 | flasky (Python) | "Let every user download all of their data (profile, posts and comments) as a single file." |
| U07 | fastapi-tdd-docker (Python) | "Replace the integer IDs that the API exposes with UUIDs, while keeping existing API clients working." |
| U08 | pallets/itsdangerous (Python) | "Change the default signing digest from SHA-1 to SHA-256." |
| U09 | aspnetcore-realworld-example-app (C#) | "Store passwords with a stronger hashing scheme without forcing existing users to reset their passwords." |
| U10 | CleanArchitecture (C#) | "Make every to-do list and item belong to a tenant, so that each company only sees its own data." |

| Tarea | Repositorio (lenguaje) | Solicitud de cambio |
|---|---|---|
| U11 | spring-petclinic (Java) | "Let pet owners book visits online, without two visits being booked for the same vet at the same time." |
| U12 | starlette (Python) | "Make the session middleware's cookie Secure and SameSite=Strict by default." |

*Nota.* Las direcciones completas de los repositorios y los identificadores de *commit* están en benchmark/study-2026-11b/tasks.json (Flores, 2026b). Las solicitudes se enviaron al agente en inglés, como están aquí; la Tabla C1 trae la traducción. *Slug* es la parte legible de la dirección de una página, UUID es un identificador largo y prácticamente único, *tenant* es cada cliente de un sistema que atiende a varios al mismo tiempo, *middleware* es el código que trata cada solicitud antes que la aplicación, y SHA-1 y SHA-256 son funciones de hash, que convierten datos en un código de longitud fija.

El *harness*, el programa que hace funcionar al modelo como agente, es Claude Code, versión 2.1.319: cada ejecución es una corrida no interactiva suya en modo de planificación, un modo en que el agente puede leer el repositorio y no puede modificarlo (Anthropic, 2026). La corrida parte de un clon nuevo del *commit* fijado, en una carpeta de configuración aislada, con toda variable de entorno heredada cuyo nombre empieza por CLAUDE o ANTHROPIC eliminada. ASP y los cinco *plugins* sincronizados de la cuenta del operador se desactivaron; los cuatro *plugins* incorporados en Claude Code quedaron activos en todas las ejecuciones, en las dos condiciones (Flores, 2026b). Los dos modelos son Claude Opus 5.5 y Claude Sonnet 5.5. Las 240 ejecuciones se pusieron en un solo orden mediante una permutación con semilla, de modo que condiciones y modelos quedan intercalados en el tiempo, y se corrieron de tres en tres.

### 4.4. Riesgos de referencia, escritos a ciegas

La lista de referencia de una tarea es el conjunto de riesgos contra el cual se puntúa toda respuesta de esa tarea, y las doce listas juntas forman la vara de medir del estudio. La escribieron tres modelos de otros proveedores: Grok, GPT-OSS 120B y Gemini 3.8 Flash (Tabla 3). Cada uno leyó una copia nueva del repositorio y la solicitud de cambio, bajo un único *prompt* que pide de seis a doce formas concretas en que el cambio puede salir mal en esa base de código (Apéndice A). Ninguno vio las condiciones, el protocolo ni salida alguna del agente probado. Produjeron 369 elementos. Un juez agrupó entonces los elementos que nombran el mismo problema, y un riesgo entró en la lista de referencia cuando al menos dos de los tres revisores lo habían citado. Quedaron 100 riesgos de referencia, entre 6 y 10 por tarea; 31 fueron citados por los tres revisores y 69 por dos (el Apéndice B muestra una lista completa).

Aquí los modelos hacen las veces de ingenieros independientes, una sustitución que las directrices para estudios con LLM piden a los autores declarar y delimitar (Baltes et al., 2027). El protocolo la declara como un debilitamiento. El *prompt* de los revisores también comparte palabras clave con la frase probada, y la Sección 7, objeción 1, lo trata como la principal amenaza al resultado.

**Tabla 3**
*Los papeles que cada modelo externo ejerce en el estudio*

| Modelo | Escribe una lista de revisor | Agrupa los elementos | Juzga la cobertura | Juzga la relevancia | Riesgos de referencia con su redacción |
|---|---|---|---|---|---|
| Gemini 3.8 Flash | sí | sí | juez principal | sí | 91 |
| GPT-OSS 120B | sí | no | no | no | 9 |
| Grok | sí | no | segundo juez | no | 0 |
| GPT-6 Astra | no | no | tercer juez, solo en la segunda vara de medir | no | 0 |

*Nota.* Un riesgo de referencia lleva la redacción del primer revisor, en orden alfabético de nombre de revisor, que lo escribió, razón por la cual ninguno lleva la redacción de Grok. Ningún modelo externo comparte proveedor con los modelos probados. Las salidas guardadas del cliente de Grok no llevan identificador de modelo. GPT-6 Astra se añadió después de la revisión, como juez no planificado de la segunda vara de medir, y es del mismo proveedor que GPT-OSS 120B (Sección 5.8; Flores, 2026b).

### 4.5. Puntuación

La *cobertura* es el desenlace: el número de riesgos de referencia que la tabla de una respuesta cubre, dividido por el número de riesgos de referencia de la tarea. El juez principal, Gemini 3.8 Flash, recibe la solicitud de cambio, los riesgos de referencia y solo las filas de la tabla de la respuesta (Apéndice A). Para cada riesgo de referencia, decide si alguna fila trata el mismo problema de fondo, aunque esté redactado de otro modo. La Figura 6 muestra los veredictos del juez para las dos respuestas de la Figura 5. Al juez no se le informa la condición ni el modelo. Una lista fija de palabras se cambió por un marcador antes del juicio; la lista se heredó del estudio de origen, en el que ocultaba el nombre del protocolo ASP, y alteró 12 palabras en 10 de las 240 tablas de aquí (Flores, 2026b).

**Figura 6**

*Cómo se puntúan las dos respuestas de la Figura 5: el veredicto del juez principal sobre cada uno de los seis riesgos de referencia de la tarea*

| Riesgo de referencia (tarea U08) | simple t078 | con la frase t051 |
|---|---|---|
| 1 Los tokens antiguos dejan de validarse tras la actualización | ✓ | ✓ |
| 2 Las dos clases acaban con valores por defecto distintos | ✗ | ✓ |
| 3 La clave derivada cambia de valor y de tamaño | ✗ | ✗ |
| 4 Las pruebas afirman salidas que coinciden con SHA-1 | ✓ | ✓ |
| 5 Las firmas más largas desbordan columnas, URL o cookies | ✓ | ✓ |
| 6 La documentación y los doctests aún dicen SHA-1 | ✗ | ✓ |
| **Cubiertos** | **3 de 6 = 50%** | **5 de 6 = 83%** |

Tres medidas secundarias se fijaron junto con ella (Flores, 2026b):

- **Segundo juez.** Grok puntúa el mismo material del mismo modo. La concordancia entre los dos jueces se reporta a nivel de decisiones individuales; el κ de Cohen (Cohen, 1960), que descuenta la concordancia esperada por azar, se añadió después a ese reporte y se rotula como no planificado (Sección 7).
- **Comparador léxico.** Un comparador determinista por superposición de tokens, que compara textos por los tokens (palabras o partes de palabras) que comparten, mantenido del estudio de origen, en el que ya se había mostrado ciego a la paráfrasis (Flores, 2026a).
- **Relevancia.** La parte de las filas de la tabla de una respuesta que el juez principal llama riesgo concreto y plausible de este cambio específico. La relevancia protege contra una mejora vacía, porque el efecto del pre-mortem humano es de cantidad (Mitchell et al., 1989). Se juzga solo a partir de la solicitud y de la fila, de modo que mide especificidad y plausibilidad, no corrección.

Los jueces LLM alcanzan más de un 80% de concordancia con las preferencias humanas en algunos escenarios (Zheng et al., 2023) y tienen sesgos conocidos, entre ellos la preferencia por texto parecido al propio (Panickssery et al., 2024). Ningún juez de este estudio comparte proveedor con los modelos probados, por lo que ningún juez puntúa respuestas escritas por un modelo de su propio proveedor, el caso en que la autopreferencia está documentada. Lo mismo no ocurre con los riesgos de referencia, la mayoría de los cuales lleva la redacción del juez principal (Tabla 3; Sección 7, objeción 3).

### 4.6. Análisis y regla de decisión

El estadístico de prueba es la media, sobre los 24 estratos, de la diferencia entre la cobertura media con la frase y la cobertura media sin ella. Su significación proviene de una prueba de permutación (Ernst, 2004): las etiquetas de condición se permutan dentro de cada estrato 10.000 veces, y el valor p es la parte de las permutaciones cuyo estadístico es al menos el observado, contada como (aciertos + 1) / 10.001 para que nunca sea cero (Phipson y Smyth, 2010). La prueba es unilateral y no supone ninguna distribución para la cobertura. Permutar dentro de los estratos mantiene las diferencias entre tareas y entre modelos fuera de la comparación (Ernst, 2004).

En símbolos, el estadístico $D$ es la media de la Ecuación 1, donde $S$ es el número de estratos y $d_s$ es la diferencia, en el estrato $s$, entre la cobertura media con la frase y la cobertura media sin ella (Flores, 2026b).

$$D = \frac{1}{S}\sum_{s=1}^{S} d_s \tag{1}$$

El valor p sigue la Ecuación 2, donde *B* es el número de permutaciones y *b* es el número de ellas cuyo estadístico es al menos el *D* observado (Phipson y Smyth, 2010). Con *B* = 10.000, el menor valor que la prueba puede devolver es 1/10.001.

$$p = \frac{b + 1}{B + 1} \tag{2}$$

La regla de decisión tiene una condición para "confirmado": valor p combinado por debajo de ,05 *y* la misma prueba dentro de cada modelo, sobre sus doce estratos, por debajo de ,05 para los dos modelos (Flores, 2026b). Si solo un modelo pasara, el resultado se declararía confirmado únicamente para ese modelo. La regla también exigía al menos 18 estratos completos. Ningún otro análisis es confirmatorio, en el sentido de Nosek et al. (2018).

Se fijó de antemano que un error en una ejecución cuenta como cobertura cero y relevancia cero y no se repite, salvo un error de límite de uso, que interrumpe la recolección (Flores, 2026b). La Sección 5.5 examina lo que hizo esa regla.

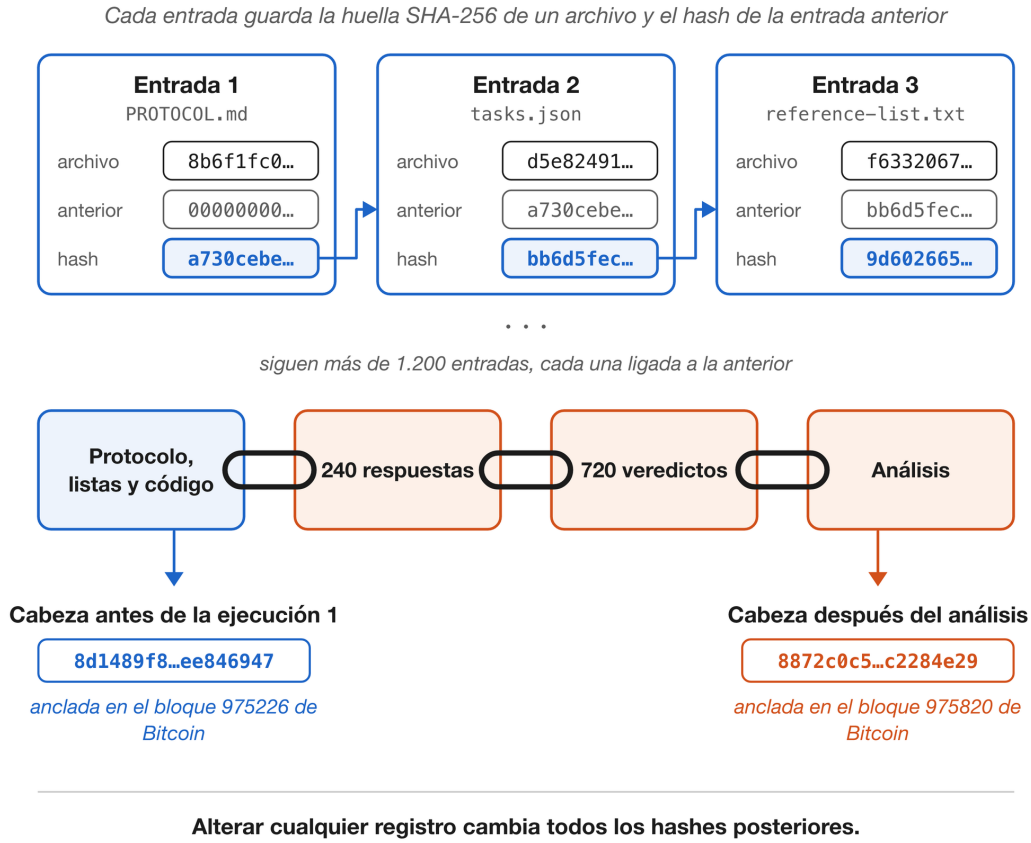
### 4.7. Prerregistro y registro de integridad

Prerregistro aquí significa que el protocolo, las tareas, los *prompts*, las listas de referencia y el código de análisis se registraron en *commits* de un repositorio público y recibieron un sello de tiempo antes de la primera ejecución contada (Nosek et al., 2018). Los añadidos posteriores que enumera la Tabla 4 se fijaron cada uno en un anexo, un texto fechado que se añade al protocolo y se sella del mismo modo antes de las llamadas que cubre. Un registro encadenado por hash (*ledger*) liga cada archivo detrás de los resultados reportados (los once veredictos parciales de la Sección 8 quedan fuera de él): cada entrada guarda la huella SHA-256 de un archivo y de la entrada anterior, de modo que el registro no puede reescribirse sin cambiar su última entrada, llamada cabeza (Flores, 2026b). Un archivo corregido después de registrado recibe una nueva entrada, y la antigua

permanece en la cadena. Una huella SHA-256 es un código de 64 caracteres calculado a partir de un archivo; cambiar un byte del archivo cambia el código. La Figura 7 muestra las tres entradas iniciales del registro con sus huellas reales y las etapas que la cadena cierra.

**Figura 7**

*La cadena de hashes del registro del estudio: las tres entradas iniciales, las dos cabezas con sello de tiempo y las etapas entre ellas*




La cabeza del registro antes de la ejecución 1 recibió un sello de tiempo de OpenTimestamps, servicio que ancla una huella en la *blockchain* de Bitcoin, y el *commit* se guardó en Software Heritage, un archivo público de código fuente. La Tabla 4 enumera cada evento sellado con el registro que prueba su orden, y la Figura 8 muestra la secuencia hasta el análisis prerregistrado. El sello y la primera ejecución están a 21 segundos uno de otro en el reloj del operador: el *commit* del sello es de las 16:08:23 y la primera respuesta de la primera ejecución es de las 16:08:44. Ese

orden de segundos se apoya en estos registros locales; el ancla en Bitcoin y el archivo muestran que los archivos sellados existían cuando se hicieron esas pruebas, no que precedieran a la primera ejecución por segundos. Las 240 respuestas se publicaron antes de que existiera cualquier veredicto, y todos los veredictos se añadieron al registro antes de que se ejecutara el análisis (Flores, 2026b).

**Tabla 4**
*El registro de integridad del estudio*

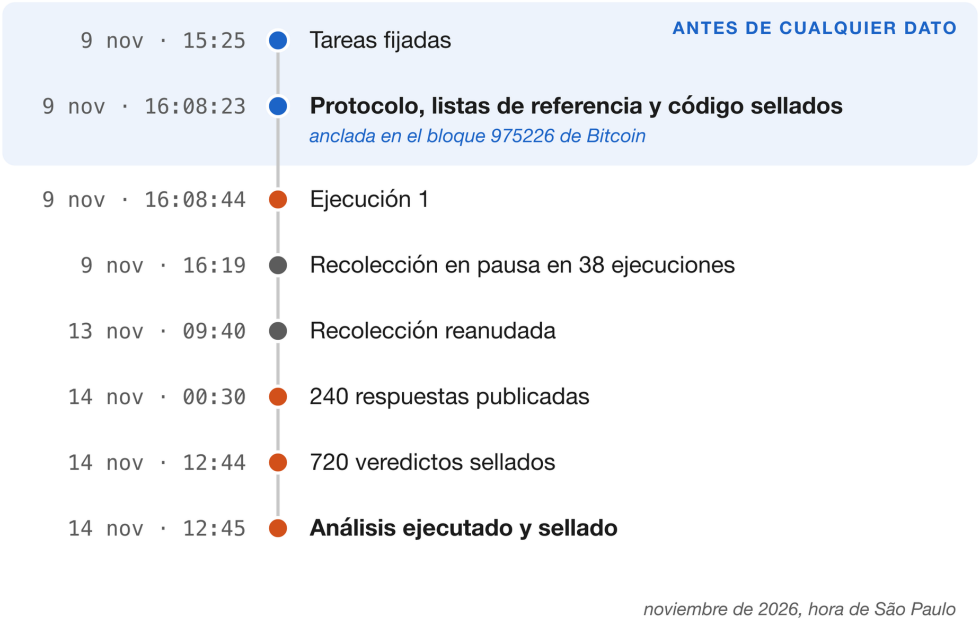
| **Evento** | **Fecha, hora de São Paulo** | **Registro** |
|---|---|---|
| Tareas fijadas, antes de cualquier lista de referencia | 9 nov 2026, 15:25 | *Commit* eb1a447 |
| Protocolo, listas de referencia, *prompts* y código sellados | 9 nov 2026, 16:08:23 | *Commit* fe18438; cabeza del registro 8d1489f8…ee846947; prueba de OpenTimestamps anclada en el bloque 975226 de Bitcoin; revisión de Software Heritage fe184385434ee3d433104549 d054f8c98ad69b32 |
| Primera respuesta de la primera ejecución contada, t091 | 9 nov 2026, 16:08:44 | Primera marca de hora en runs/envelopes/ t091.jsonl.gz |
| Las 240 respuestas publicadas, antes de cualquier veredicto | 14 nov 2026, 00:30 | *Commit* e046f36 |
| Los 720 veredictos sellados, antes del análisis | 14 nov 2026, 12:44 | *Commit* 67b52d0 |
| Análisis prerregistrado ejecutado y sellado | 14 nov 2026, 12:45 | *Commit* e8c73c1; cabeza del registro 8872c0c5…c2284e29; prueba de OpenTimestamps anclada en el bloque 975820 de Bitcoin |
| Segunda vara de medir: anexo, *prompt* y código sellados antes de escribirse sus listas | 14 nov 2026, 16:25 | *Commits* 573b8bf y fa3d0ba; cabeza del registro cf0a7bfd…0499f47f; prueba de OpenTimestamps anclada en el bloque 975834 de Bitcoin |

| Evento | Fecha, hora de São Paulo | Registro |
|---|---|---|
| Segundo anexo (repetición y tercer juez) sellado antes de la primera llamada | 15 nov 2026, 11:19 | *Commits* 0acd091 y 2fb3c8b; cabeza del registro 73e275d1…58b42f11; prueba de OpenTimestamps enviada |
| Tercer anexo (juez por API de pago) sellado antes de la primera llamada de puntuación | 15 nov 2026, 14:23 | *Commits* 2b2af4c y 2c5f5eb; cabeza del registro 2a9ac120…4f6dce4c; prueba de OpenTimestamps enviada |

*Nota.* Los *commits* están en el repositorio de Flores (2026b). Las cabezas del registro están abreviadas; los valores completos están en benchmark/study-2026-11b/pretrial-head.txt y analysis-head.txt, junto a sus archivos de prueba .ots. Las pruebas pueden leerse con ots info y comprobarse contra un nodo de Bitcoin con ots verify; la tercera cabeza está en yardstick2/head-before-lists.txt. Las cabezas de los dos anexos posteriores están en addendum2/head-before-addendum2.txt y addendum3/head-before-addendum3.txt; sus pruebas se habían enviado y aún no estaban ancladas cuando se escribió esta versión.

**Figura 8**

*Orden de los eventos sellados del estudio, del archivo de tareas al análisis (eje ordinal, fuera de escala)*



# 5. Resultados

## 5.1. Recolección

Las 240 ejecuciones se registraron, en una sola versión de Claude Code, y ninguna alteró un archivo del proyecto (Flores, 2026b). Tres ejecuciones terminaron en error de red y no dejaron tabla (Sección 5.5). Los 24 estratos están completos, de modo que la regla de decisión de la Sección 4.6 se aplica (Tabla 5).

## 5.2. Resultado principal

La frase elevó la cobertura de 46,1% a 57,5% según el juez principal (Tabla 5). El estadístico prerregistrado, la diferencia media sobre los 24 estratos, es de 11,4 puntos porcentuales, con $p <$ ,001: el valor es 1/10.001, el menor que la prueba puede devolver (Phipson y Smyth, 2010). Ninguna de las 10.000 permutaciones alcanzó la diferencia observada; la mayor dio 5,9 puntos (Figura 9). Dentro de Claude Opus 5.5 la diferencia es de 10,4 puntos y dentro de Claude Sonnet 5.5 es de 12,4 puntos, cada una con $p <$ ,001. La diferencia es positiva en 22 de los 24 estratos, cero en uno y negativa en uno (Tabla 6). La regla de decisión devuelve *confirmado*.

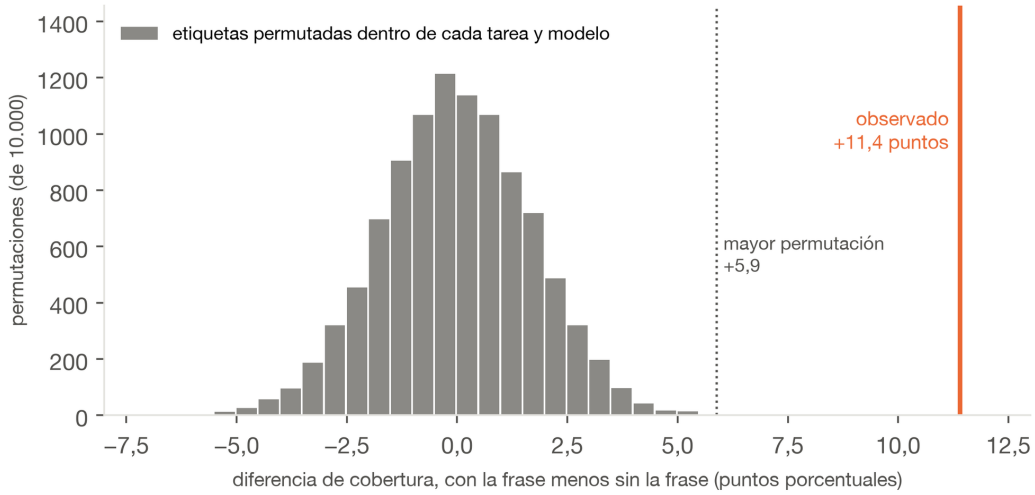
**Tabla 5**
*Cobertura de riesgos por modelo y juez*

| Modelo | Simple (juez principal) | Frase (juez principal) | Simple (segundo juez) | Frase (segundo juez) |
|---|---|---|---|---|
| Claude Opus 5.5 | 43% | 54% | 48% | 57% |
| Claude Sonnet 5.5 | 49% | 61% | 52% | 64% |
| Ambos | 46% | 58% | 50% | 61% |

*Nota.* 60 ejecuciones por modelo y condición. El juez principal es Gemini 3.8 Flash y el segundo juez es Grok. Los porcentajes se redondean a números enteros; las diferencias del texto provienen de los valores sin redondear.

**Figura 9**

*La prueba de permutación del resultado principal: las 10.000 diferencias permutadas y la observada*




Un *bootstrap* por percentiles, que recalcula la diferencia en muchos remuestreos aleatorios de los datos, no estaba previsto en el protocolo y se reporta solo como descripción. Remuestreando réplicas dentro de cada celda, da un intervalo del 95% de 8,9 a 14,0 puntos para la diferencia combinada, de 6,8 a 14,1 para Opus 5.5 y de 8,8 a 16,2 para Sonnet 5.5 (Flores, 2026b). El intervalo refleja la variación entre ejecuciones en estas doce tareas, no entre tareas.

La Figura 10 y la Tabla 6 muestran el resultado tarea por tarea. La mejora no depende de pocas tareas. El único estrato negativo es el cambio de *slug* en Go con Sonnet 5.5 (U02, −5 puntos), y el estrato en cero es la exportación de datos con Sonnet 5.5 (U06), en el que las dos condiciones ya cubrían el 73% (Tabla 6).

**Figura 10**

*Fracción de los riesgos de referencia ciegos cubierta por la tabla final de riesgos, por tarea y modelo, con y sin la frase (juez principal; leyenda como en la Figura 3)*

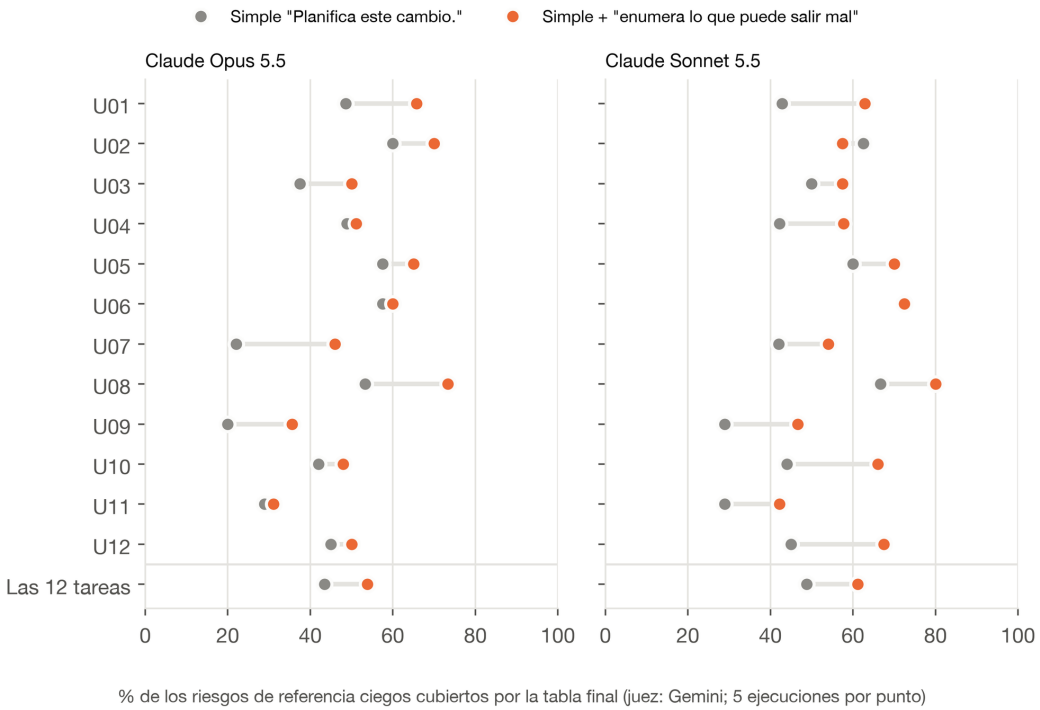



**Tabla 6**

*Cobertura de riesgos por tarea y modelo, juez principal, en porcentaje*

| Tarea | Riesgos de referencia | Opus 5.5 simple | Opus 5.5 frase | Sonnet 5.5 simple | Sonnet 5.5 frase |
|---|---|---|---|---|---|
| U01 | 7 | 49 | 66 | 43 | 63 |
| U02 | 8 | 60 | 70 | 63 | 58 |
| U03 | 8 | 38 | 50 | 50 | 58 |
| U04 | 9 | 49 | 51 | 42 | 58 |
| U05 | 8 | 58 | 65 | 60 | 70 |
| U06 | 8 | 58 | 60 | 73 | 73 |
| U07 | 10 | 22 | 46 | 42 | 54 |
| U08 | 6 | 53 | 73 | 67 | 80 |
| U09 | 9 | 20 | 36 | 29 | 47 |
| U10 | 10 | 42 | 48 | 44 | 66 |

| Tarea | Riesgos de referencia | Opus 5.5 simple | Opus 5.5 frase | Sonnet 5.5 simple | Sonnet 5.5 frase |
|---|---|---|---|---|---|
| U11 | 9 | 29 | 31 | 29 | 42 |
| U12 | 8 | 45 | 50 | 45 | 68 |

*Nota.* Cada celda es la media de cinco ejecuciones. Tres celdas simples contienen una ejecución que terminó en error de red y cuenta como cero: U07 y U09 en Opus 5.5 y U10 en Sonnet 5.5. En las otras cuatro ejecuciones, esas celdas son 28, 25 y 55.

### 5.3. Segundo juez

El segundo juez dio 50,0% sin la frase y 60,8% con ella: diferencia media de 10,7 puntos sobre los estratos ($p < ,001$), 9,3 puntos dentro de Opus 5.5 y 12,1 dentro de Sonnet 5.5 (cada uno con $p < ,001$; Tabla 5), positiva en 22 de los 24 estratos también según este juez (Flores, 2026b). Marca un poco más riesgos como cubiertos que el juez principal en las dos condiciones y llega a la misma decisión.

Los dos jueces tomaron 2.000 decisiones cada uno, una por riesgo de referencia por ejecución, y coincidieron en el 93,2% de ellas. El κ de Cohen, presentado en la Sección 4.5 (Cohen, 1960), es ,86. De esas decisiones, 29 se refieren a las tres ejecuciones sin tabla, en las que los dos jueces solo podían responder “no cubierto”; en las otras 1.971 la concordancia es del 93,1% y el κ es ,86. El comparador léxico marcó 3,0% y 1,3%; como en el estudio de origen, pasa por alto las coincidencias parafraseadas y no se usa aquí como evidencia de cobertura (Flores, 2026a).

### 5.4. Relevancia

En las ejecuciones que produjeron tabla, la parte de filas que el juez principal llamó concretas y plausibles fue de 71,1% sin la frase y de 73,9% con ella (Tabla 7). Según la regla fijada, que puntúa como cero las tres ejecuciones sin tabla, la cifra simple es de 69,4%. No se observó reducción en

la relevancia media juzgada, que es la lectura fijada en el protocolo. El estudio no fija un margen de no inferioridad, la mayor caída de relevancia que aún contaría como ausencia de pérdida, de modo que esto no prueba que la mejora de cobertura llegara sin filas más vagas, el riesgo que señala la literatura humana (Mitchell et al., 1989). No se afirma un aumento de la relevancia: la diferencia combinada de 4,5 puntos tiene p = ,011 según la regla fijada, pero ninguno de los modelos llega por sí solo a ,05 (,052 y ,060), y la diferencia no es significativa cuando se dejan de lado las ejecuciones con error (Tabla 7).

Las tablas tenían 9,2 filas sin la frase y 9,6 con ella, frente a un tope de diez (Tabla 8). La cobertura es una medida de *recall*, la parte de los elementos de referencia que una respuesta encuentra, de modo que una tabla más larga tiene más posibilidades de coincidir con un riesgo de referencia. Restringir la comparación a las ejecuciones cuya tabla tiene las diez filas completas, 56 simples y 82 con la frase, deja la diferencia en 11,3 puntos sobre los 17 estratos en que las dos condiciones permanecen (Flores, 2026b). El tamaño de la tabla es él mismo un efecto de la solicitud, de modo que esta comparación restringida describe y no aísla una causa; muestra que la mejora está presente entre tablas del mismo tamaño.

### 5.5. Sensibilidad a las tres ejecuciones con error

Durante una pérdida breve de acceso a internet, tres ejecuciones terminaron con error tras diez nuevos intentos y no produjeron tabla (Flores, 2026b). Según la regla fijada de antemano, cuentan como cero (Sección 4.6). Las tres pertenecen a la condición simple, de modo que la regla desplaza la diferencia principal hacia la hipótesis. El protocolo no lo había previsto, y una verificación etiquetada como no planificada repite el análisis de dos maneras (Tabla 7).

**Tabla 7**
*El análisis principal con y sin las tres ejecuciones con error de red*

| Análisis | Estratos | Juez principal | Segundo juez | Relevancia |
|---|---|---|---|---|
| Como se prerregistró (los errores cuentan cero) | 24 | +11,4 (p < ,001) | +10,7 (p < ,001) | +4,5 (p = ,011) |
| Sin los tres estratos que tienen un error | 21 | +10,1 (p < ,001) | +8,9 (p < ,001) | +2,5 (p = ,091) |
| Solo sin las tres ejecuciones | 24 | +10,5 (p < ,001) | +9,7 (p < ,001) | +2,9 (p = ,054) |

*Nota.* Diferencias en puntos porcentuales, frase menos simple. Solo la primera fila es confirmatoria.

El resultado de cobertura se mantiene en todas las filas de la Tabla 7, y dentro de cada modelo: en la fila más conservadora, las diferencias por modelo son de 8,5 puntos para Opus 5.5 (p < ,001) y de 11,6 para Sonnet 5.5 (p < ,001) según el juez principal, y de 6,9 (p < ,001) y 10,7 (p < ,001) según el segundo. La diferencia de relevancia no se mantiene al nivel de ,05 sin las ejecuciones con error, razón por la cual la Sección 5.4 afirma solo que no se observó reducción en la relevancia.

El límite opuesto da la misma lectura. Si cada una de las tres ejecuciones con error se puntúa como si su tabla hubiera cubierto todos los riesgos de referencia, la diferencia es de 8,9 puntos según el juez principal y de 8,2 según el segundo (Flores, 2026b).

Las tres ejecuciones también se lanzaron una vez más, dos días después de la recogida, como añadido no planificado, fijado en un anexo fechado antes de lanzarlas (Sección 8). Cada una terminó con normalidad, con tabla de 8, 9 y 10 filas. Con su cobertura en lugar de los tres ceros, la diferencia en las listas originales es de 10,5 puntos (p < ,001): 9,0 en Opus 5.5 y 11,9 en Sonnet 5.5 (Flores, 2026b). Las filas fijadas de antemano no cambian, y la Sección 5.8 da los límites de esta repetición.

### 5.6. Costo

El tiempo medio de ejecución fue menor con la frase, igual que los turnos y los tokens de salida, en los dos modelos (Tabla 8). El protocolo enumera estas medidas como descriptivas y no les asigna prueba, de modo que se reportan como descripción (Flores, 2026b). Sostienen una única afirmación: no se observó costo adicional.

Una diferencia de comportamiento está detrás de parte de esos números. En el modo de planificación, el agente puede guardar el plan en un archivo, con una llamada a una herramienta, antes de responder (Anthropic, 2026). En Opus 5.5 lo hizo en 59 de las 60 ejecuciones simples y en 22 de las 60 ejecuciones con la frase; en Sonnet 5.5 no lo hizo nunca, en ninguna condición (Flores, 2026b). La Sección 7, objeción 13, examina si esa diferencia podría explicar la mejora de cobertura. Las dos ejecuciones de Opus 5.5 que terminaron en error de red (Sección 5.5) ya habían guardado un archivo de plan cuando cayó la conexión, y la de Sonnet 5.5 no. Una ejecución simple en Opus 5.5 duró de media un turno más (Tabla 8) y estuvo más tiempo expuesta a la caída.

**Tabla 8**

*Costo medio por ejecución, por condición y modelo, en las ejecuciones que no terminaron en error*

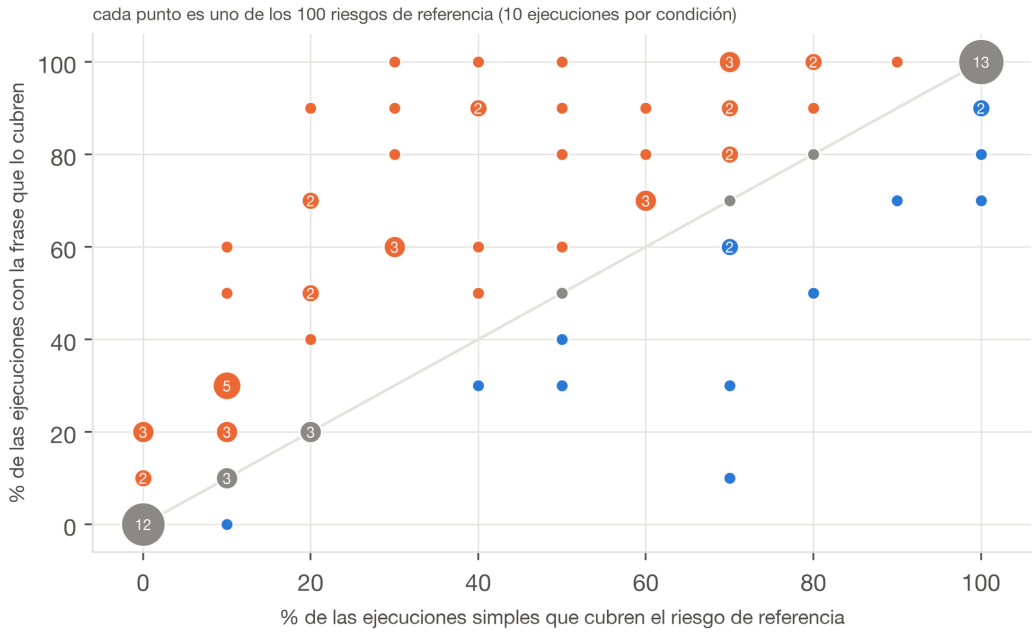
| Modelo | Condición | Segundos | Turnos | Tokens de salida | Filas de la tabla |
|---|---|---|---|---|---|
| Claude Opus 5.5 | Simple | 66,5 | 5,8 | 5.801 | 9,2 |
| Claude Opus 5.5 | Frase | 53,5 | 4,8 | 4.736 | 9,4 |
| Claude Sonnet 5.5 | Simple | 32,9 | 3,5 | 3.014 | 9,3 |
| Claude Sonnet 5.5 | Frase | 27,3 | 3,2 | 2.469 | 9,7 |
| Ambos | Simple | 49,6 | 4,7 | 4.395 | 9,2 |
| Ambos | Frase | 40,4 | 4,0 | 3.603 | 9,6 |

*Nota.* Un turno es un intercambio entre el agente y sus herramientas o el modelo. 117 ejecuciones simples y 120 ejecuciones con la frase.

### 5.7. Qué riesgos añade la frase (exploratorio)

Esta sección no estaba planificada y no prueba nada. Examina los 100 riesgos de referencia uno por uno, según el juez principal, para describir de dónde viene la mejora (Figura 11). En la figura, un punto naranja por encima de la diagonal es un riesgo cubierto más veces con la frase, un punto gris sobre ella es un riesgo cubierto el mismo número de veces y un punto azul por debajo es un riesgo cubierto menos veces; el número dentro de un punto cuenta los riesgos en esa posición.

**Figura 11**
*Cada riesgo de referencia según la parte de ejecuciones que lo cubren, sin y con la frase (juez principal)*




- **Amplitud.** 52 riesgos de referencia se cubrieron con más frecuencia con la frase, 34 con la misma frecuencia y 14 con menos. Cada riesgo se puntuó en diez ejecuciones por condición, de modo que cada una de estas diferencias es de al menos diez puntos (Figura 11).

- **Confiabilidad.** El número de riesgos de referencia que la mayoría de las ejecuciones cubrió subió de 41 a 56. La frase transformó sobre todo riesgos que el agente citaba a veces en riesgos que citaba la mayor parte de las veces (Figura 11).
- **Puntos ciegos que permanecen.** 12 riesgos de referencia no fueron cubiertos por ninguna ejecución en ninguna de las condiciones, y 13 por ninguna ejecución con la frase. Son sobre todo consecuencias estrechas, a nivel de archivo, como un índice compuesto de base de datos ausente o un formato de almacenamiento en que no cabría un nuevo esquema de hash (Flores, 2026b).

Las mayores mejoras son riesgos de interacción entre el cambio y lo que ya existe: una ruta de registro existente que inicia sesión al usuario de inmediato y así elude la nueva verificación de correo (U01, de 30% a 100% de las ejecuciones), filas existentes que quedan sin valor cuando se añade una nueva columna de identificador (U07, de 30% a 90%) y un generador síncrono de hash de contraseñas que un algoritmo de uso intensivo de memoria bloquearía (U09, de 20% a 90%). Las mayores pérdidas están en U02 y U03, en riesgos que el plan simple ya citaba en la mayoría de las ejecuciones (Flores, 2026b). El Apéndice B muestra la misma lectura para una lista completa.

### 5.8. Una segunda vara de medir, con otras palabras (exploratorio)

Esta comprobación se especificó después de conocerse los resultados anteriores y no pone a prueba nada que estuviera prerregistrado. Responde a la objeción 1 de la Sección 7: las listas de referencia se pidieron con palabras próximas a las de la frase puesta a prueba. Los tres revisores escribieron listas nuevas para las doce tareas, con un *prompt* en el que cambió una frase. Pide “the main risks of this change” (los principales riesgos de este cambio), que son las palabras del contrato de salida

que reciben las dos condiciones, y las palabras “go wrong” (salir mal) no aparecen (Flores, 2026b). La regla que fusiona las listas, las 240 respuestas, el *prompt* del juez y el estadístico quedaron como estaban. El anexo, el *prompt* y el código se sellaron en el registro y recibieron sello de tiempo antes de llamar al primer revisor (Tabla 4).

Las listas nuevas tienen 111 riesgos de referencia, entre 7 y 11 por tarea. Frente a ellas el juez principal da 55,8% sin la frase y 62,5% con ella: una diferencia media de 6,7 puntos ($p < ,001$), 6,5 en Opus ($p = ,002$) y 6,9 en Sonnet ($p = ,001$). De los 24 estratos, 16 son positivos, 2 son cero y 6 son negativos (Tabla 9).

**Tabla 9**
*Cobertura de riesgos frente a las dos varas de medir, juez principal*

| **Vara de medir, por las palabras de su *prompt*** | **Riesgos de referencia** | **Sin la frase** | **Con la frase** | **Diferencia** | **p** | **Estratos positivos** |
|---|---|---|---|---|---|---|
| Original: “ways this change could go wrong” (formas en que este cambio puede salir mal) | 100 | 46,1% | 57,5% | +11,4 | < ,001 | 22 de 24 |
| Segunda: “the main risks of this change” (los principales riesgos de este cambio) | 111 | 55,8% | 62,5% | +6,7 | < ,001 | 16 de 24 |

*Nota.* Las diferencias están en puntos porcentuales, como media sobre los 24 estratos de tarea por modelo; p procede de la prueba de permutación estratificada de la Sección 4.6. La segunda fila es exploratoria y se calculó después de conocerse la primera.

La diferencia sigue siendo positiva cuando la petición a los revisores deja de usar las palabras de la frase, lo que el anexo, sellado antes de escribirse las listas, leía como una ganancia no atada a esa redacción (Flores, 2026b). Esto no excluye un efecto de la redacción en los pasos posteriores que construyen la vara de medir, descritos abajo. La diferencia también es menor. 6,7 puntos queda por debajo del intervalo *bootstrap* de la estimación original, de 8,9 a 14,0 (Sección 5.2), y el efecto es menos uniforme entre las tareas (Tabla 9). Con las listas nuevas, la solicitud simple gana unos

diez puntos de cobertura y la solicitud con la frase gana cinco. La objeción 1 queda, por tanto, aceptada en parte.

Los datos no dicen cuánto de la distancia entre 11,4 y 6,7 procede de las palabras. Esta es una sola obtención nueva de listas; los dos conjuntos de listas difieren en tamaño y en contenido, y una cobertura inicial más alta deja menos margen de ganancia (Tabla 9). Tres límites valen para esta comprobación. La petición que fusiona las tres listas se mantuvo como estaba y todavía las describe como “ways a software change could go wrong”; la comprobación, por tanto, quita esas palabras de la petición a los revisores, y no de toda la construcción de la vara de medir (Flores, 2026b). Gemini escribió la redacción de 98 de los 111 riesgos de referencia y fue su único juez en el análisis planificado, la concentración que describe la objeción 3 de la Sección 7. El segundo juez planificado alcanzó su límite de uso tras 11 de 240 veredictos; el anexo prohíbe sustituir a un juez, de modo que sus veredictos no están en esta versión (Sección 8).

Otras tres comprobaciones de esta vara de medir se pidieron en la revisión ciega de este manuscrito y se añadieron después de conocerse su resultado; ninguna está en el anexo (Sección 8). Las tres ejecuciones con error de red son todas ejecuciones simples y también aquí cuentan cero. Sin los tres estratos que las contienen, la diferencia es de 5,3 puntos ($p < ,001$), y sin las tres ejecuciones es de 5,6 ($p < ,001$). En el límite opuesto, con cada ejecución con error puntuada como cobertura total, es de 4,2 puntos ($p = ,005$): 5,3 en Sonnet 5.5 ($p = ,010$) y 3,2 en Opus 5.5, donde deja de ser significativa ($p = ,099$). La lectura fijada en el anexo se apoya en la prueba conjunta y se mantiene en todos los tratamientos; el resultado aislado de Opus 5.5 no se mantiene en ese límite (Flores, 2026b). El intervalo *bootstrap* de la segunda diferencia va de 4,1 a 9,4 puntos y se solapa con el intervalo original, de 8,9 a 14,0. Remuestrear las doce tareas, y no las réplicas, lo ensancha a 2,2 a 11,1, frente a 8,2 a 14,5 en la vara de medir original.

Dos añadidos llegaron después de la segunda ronda de revisión, cada uno fijado en un anexo fechado, sellado y con sello de tiempo antes de la primera llamada (Sección 8). Primero, las tres ejecuciones con error se lanzaron una vez más, un intento cada una, con la misma petición, el mismo *commit* fijado, el mismo identificador de modelo y la misma versión de Claude Code; las tres terminaron con normalidad (Sección 5.5). Segundo, GPT-6 Astra, modelo de un tercer proveedor, puntuó las 240 respuestas y las tres repeticiones frente a la segunda vara de medir, con la petición del juez sin cambios, como juez no planificado (Tabla 10). No sustituye al segundo juez planificado, cuyos veredictos siguen pendientes (Flores, 2026b).

**Tabla 10**

*La diferencia en la segunda vara de medir bajo tres tratamientos de las ejecuciones con error, por juez*

| Tratamiento de las tres ejecuciones con error | Juez | Diferencia | Opus 5.5 | Sonnet 5.5 |
|---|---|---|---|---|
| Cuentan cero, como se fijó de antemano | Gemini 3.8 Flash | +6,7 (p < ,001) | +6,5 (p = ,002) | +6,9 (p = ,001) |
| Cuentan cero, como se fijó de antemano | GPT-6 Astra | +9,0 (p < ,001) | +8,1 (p < ,001) | +9,8 (p < ,001) |
| Puntuadas como cobertura total | Gemini 3.8 Flash | +4,2 (p = ,005) | +3,2 (p = ,099) | +5,3 (p = ,010) |
| Puntuadas como cobertura total | GPT-6 Astra | +6,5 (p < ,001) | +4,7 (p = ,013) | +8,2 (p < ,001) |
| Lanzadas una vez más | Gemini 3.8 Flash, por API | +5,6 (p < ,001) | +4,9 (p = ,009) | +6,4 (p = ,002) |
| Lanzadas una vez más | GPT-6 Astra | +7,7 (p < ,001) | +6,2 (p < ,001) | +9,1 (p < ,001) |

*Nota.* Diferencias en puntos porcentuales, frase menos simple, con el p de permutación de la Sección 4.6. Todas las filas son exploratorias. En la quinta fila, los veredictos de Gemini sobre las tres repeticiones llegaron por una API de pago, no por el cliente de línea de comandos de los 240 veredictos sellados.

Todas las filas de la Tabla 10 son positivas, y la prueba conjunta queda por debajo de ,05 en cada una. Según GPT-6 Astra, la diferencia bajo la regla fijada es de 9,0 puntos, del 64,0% al 73,0%, con 20 de 24 estratos positivos, y sigue siendo significativa en cada modelo en el límite opuesto,

donde el resultado aislado de Opus 5.5 según Gemini no se mantenía. Con las ejecuciones lanzadas una vez más, los dos jueces dan una diferencia significativa en los dos modelos. Los dos jueces coinciden en el 89,4% de 2.192 decisiones individuales (κ de Cohen = ,77), y GPT-6 Astra marca más ítems como cubiertos, el 68,8% frente al 59,0% (Flores, 2026b).

Estos añadidos tienen límites propios. La repetición se hizo dos días después de la recogida, por un operador sin cegamiento respecto al resultado, y nada observable prueba que el modelo detrás de un identificador no haya cambiado; el modelo de la sesión, los modelos usados y los *plugins* que declaran las tres sesiones son los de las ejecuciones selladas. GPT-6 Astra es del mismo proveedor que GPT-OSS 120B, que escribió una de las tres listas de revisor. El juez principal alcanzó su límite de uso en la línea de comandos tras 4 de sus 9 veredictos sobre las repeticiones; por decisión del autor, registrada en un tercer anexo, los 9 los produjo el mismo modelo por una API de pago en lugar de esperar a que se renovara el límite, una desviación más. Una muestra de 30 ejecuciones selladas, fijada por semilla y puntuada por ese canal, coincide con los veredictos sellados en el 98,2% de 281 decisiones (κ = ,96), con una cobertura media del 56,0% frente al 56,4%. Los veredictos de la línea de comandos se reportarán junto a estos cuando se renueve el límite (Flores, 2026b).

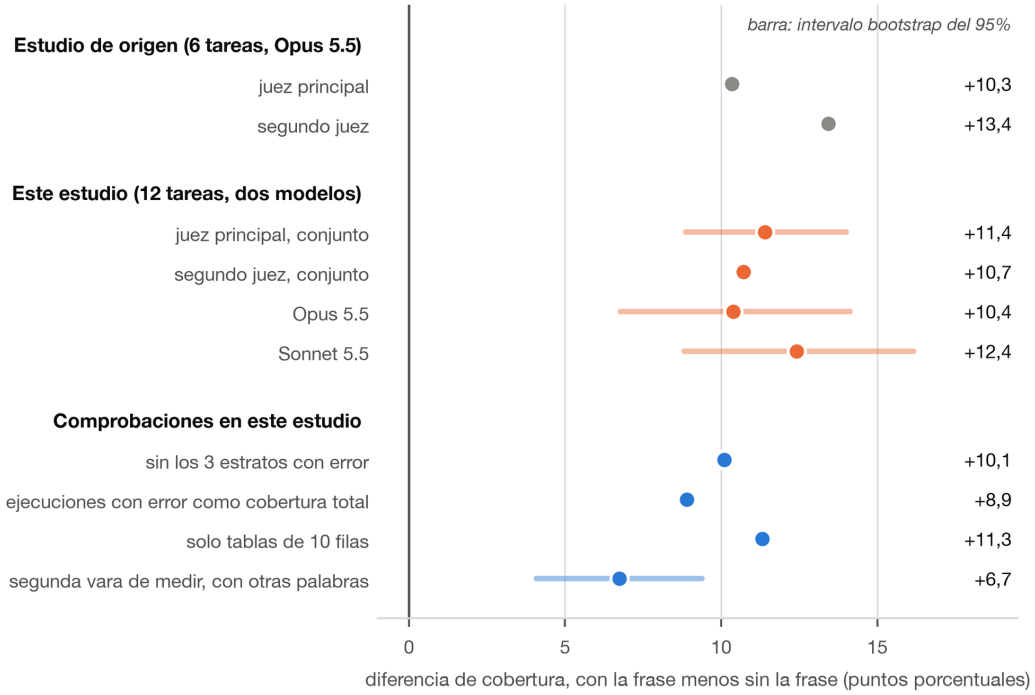
## 6. Discusión

**Lo que se confirmó.** Pedir a un agente de código las formas en que un cambio puede salir mal, antes de que planifique, hace que su lista final de riesgos cubra más de lo que esperaban tres revisores independientes (Tabla 5). El efecto se replicó en un segundo estudio, con doce tareas nuevas y un segundo modelo, en un tamaño próximo al del estudio de origen: +10 puntos allí (Tabla 1), +11 aquí. Frente a una segunda vara de medir, pedida con otras palabras, la ganancia es menor, 6,7 puntos, y sigue siendo positiva (Sección 5.8). Es modesto en términos absolutos: el

agente aún deja de citar unos cuatro riesgos de referencia de cada diez. La Figura 12 pone lado a lado las principales estimaciones de la diferencia; los añadidos posteriores están en la Tabla 10.

**Figura 12**

*Las principales estimaciones de la diferencia de cobertura: del estudio de origen, de este estudio y de sus verificaciones*




**Una lectura posible.** Las dos condiciones terminan con la misma exigencia de una tabla de riesgos, de modo que al agente se le pregunta por el riesgo en ambas. Lo que difiere es que los modos de fallo se piden antes y con otras palabras. Eso concuerda con la explicación ofrecida para InvThink, la de que la enumeración saca a la luz conocimiento que una solicitud directa deja sin uso (Kim et al., 2026), y con el hallazgo exploratorio de que la frase sobre todo vuelve regulares riesgos ocasionales (Figura 11). El estudio no prueba esa lectura. Una ablación (una repetición con un solo elemento cambiado) que moviera la misma frase a después del plan la probaría.

**El compromiso entre procedimiento y cobertura.** En el estudio de origen, un protocolo de 13 fases cubrió menos riesgos de referencia en su tabla final que esta frase, con tres veces y media el

tiempo de ejecución (Tabla 1). Más procedimiento compró menos cobertura. Los dos estudios no muestran que la estructura sea inútil; los *pipelines* multiagente añaden modos de fallo propios (Cemri et al., 2025). Muestran que un protocolo estructurado debe medirse contra la frase, que es la línea de base barata, y que ser mejor que una solicitud simple no basta.

**Los planes no son código.** Un *prompt* que pide a un agente de código que prevea casos límite elevó la tasa conjunta de tareas resueltas que evitan un defecto posterior en un modelo Claude y la redujo en otro (Patel et al., 2026). Un plan que nombra un riesgo no garantiza código que lo trate, y nada en la Tabla 5 dice lo contrario.

**Para la práctica.** La intervención es una frase al comienzo de una solicitud de planificación, y no se observó costo adicional en estas ejecuciones (Tabla 8). Un equipo que use un agente de código en modo de planificación puede adoptarla sin herramientas adicionales. Lo que el equipo no debe concluir es que el plan resultante sea seguro: su tabla nombró el 58% de los riesgos de referencia, el trabajo humano de revisión que queda no se midió, y los doce riesgos que ninguna ejecución nombró (Sección 5.7) recuerdan que los puntos ciegos del agente y la atención de sus revisores no se solapan por completo.

## 7. Objeciones y amenazas a la validez

Esta sección toma las objeciones una a una. Siguen las clases habituales de amenazas en la ingeniería de software empírica (Wohlin et al., 2012) y las cautelas planteadas para estudios que usan LLM (Sallou et al., 2024; Baltes et al., 2027). Antes de escribirse esta versión, el manuscrito y sus datos se entregaron a dos modelos de otros proveedores, cada uno instruido, de forma independiente, para actuar como revisor hostil; los dos plantearon la objeción 1. A uno de ellos se le dio luego la propia frase en estudio, dirigida al manuscrito: antes de publicar este artículo,

enumera las formas en que puede salir mal. Esa solicitud produjo la objeción 13, que ninguna de las dos revisiones había planteado. El repositorio guarda los tres informes (Flores, 2026b).

**1. La vara de medir se obtuvo con las palabras de la frase probada (constructo).** A los revisores que escribieron las listas de referencia se les indicó dar "the concrete ways this change could go wrong" (Apéndice A). La condición probada pide al agente que "list the ways this change could go wrong", y la condición simple nunca ve esas palabras. Parte de la mejora podría ser, por tanto, una coincidencia entre las respuestas probadas y la pregunta que produjo la vara de medir, y no más de los riesgos reales del cambio. Las dos condiciones se puntúan contra las mismas listas, lo que protege la comparación de diferencias entre listas; no la protege de un tratamiento que conduce al agente hacia el marco de los revisores. Nada en los datos prerregistrados separa las dos lecturas. La medida de relevancia, que no usa las listas de referencia, no muestra reducción en la relevancia media juzgada de las filas (Sección 5.4), y eso es solo una respuesta parcial. Una segunda vara de medir, obtenida con las palabras del contrato de salida que reciben las dos condiciones, se especificó después de conocerse estos resultados, en un anexo fechado sellado antes de su recolección (Flores, 2026b). Frente a ella la diferencia es de 6,7 puntos ($p < ,001$), positiva en los dos modelos y unas cuatro décimas partes menor que la estimación prerregistrada (Sección 5.8). Con los otros tratamientos de las ejecuciones con error y según un juez de un tercer proveedor va de 4,2 a 9,0 puntos, y solo su valor aislado en Opus 5.5, en el límite opuesto, no es significativo (Tabla 10). La objeción queda aceptada en parte: la ganancia no desaparece cuando la petición a los revisores deja de usar las palabras de la frase, aunque la petición que fusiona las listas conserva palabras afines (Sección 5.8), y su tamaño sí depende de la vara de medir.

**2. Las listas de referencia las escribieron modelos (constructo).** La cobertura mide la concordancia con lo que tres LLM de otros proveedores esperaban que saliera mal, filtrada a los

elementos que al menos dos de ellos citaron (Sección 4.4). No es una medida de riesgo real. Un riesgo que los tres revisores pasaron por alto no puede acreditarse. Tampoco un riesgo que solo uno de ellos nombró: la fusión conserva un riesgo cuando al menos dos revisores lo nombran, lo que deja fuera 132 de los 232 grupos de las listas originales y 102 de los 213 de la segunda (Flores, 2026b). Ese filtro puede favorecer los riesgos que los modelos tienden a repetir frente a los que solo vio un revisor, y ningún humano ha auditado los grupos que quedaron fuera. En el experimento de campo de Caraeni et al. (2026), los planificadores humanos captaron riesgos específicos del contexto que el modelo trabajando solo no captó, de modo que una vara de medir humana podría cambiar los valores absolutos. El paso (i) de la Sección 9 trata de eso. La cobertura da además el mismo peso a todo riesgo de referencia, sea cual sea la gravedad del fallo que nombra.

**3. Un modelo ocupa cuatro papeles (constructo).** Gemini 3.8 Flash escribió una lista de revisor, agrupó los elementos, es el juez principal de la cobertura y el único juez de la relevancia, y 91 de los 100 riesgos de referencia llevan su redacción (Tabla 3). Un juez puede coincidir con más facilidad con su propia redacción (Panickssery et al., 2024). El segundo juez es el control para eso: Grok no escribió ninguna redacción de referencia, y halla una mejora de 10,7 puntos frente a 11,4 (Tabla 7). La relevancia no tiene tal control, lo que es una segunda razón para afirmar solo que no se observó reducción. En la segunda vara de medir, donde 98 de los 111 riesgos llevan la redacción de Gemini, un juez de un tercer proveedor encuentra una ganancia mayor que la de Gemini (Tabla 10).

**4. El desenlace es la tabla final (constructo).** El juez ve solo las filas de la tabla. Un agente que enumera antes los modos de fallo puede simplemente copiar esa lista a la tabla, y el estudio no midió si el plan en sí trata los riesgos, si los riesgos son verdaderos para la base de código, ni si el código escrito a partir del plan tiene menos defectos. El estudio más cercano con desenlace en

código halló resultados que dependen del modelo (Patel et al., 2026). La afirmación es sobre lo que nombra la tabla de riesgos del plan.

**5. El diseño no separa posición de redacción (interna).** La frase difiere de la solicitud simple en lo que pide y en cuándo lo pide (Apéndice A). La Sección 6 ofrece el orden como una lectura y no la prueba. La ablación es el paso (iii) de la Sección 9.

**6. Las tres ejecuciones con error favorecen la hipótesis (interna).** Las tres caen en la condición simple y cuentan como cero según la regla fijada. Sin ellas el resultado se mantiene (Tabla 7; Sección 5.5).

**7. Las tablas más largas podrían explicar un *recall* mayor (interna).** Las tablas con la frase tienen en promedio 0,4 filas más (Tabla 8). Entre las ejecuciones con las diez filas completas, la diferencia es de 11,3 puntos (Sección 5.4).

**8. El juez podría percibir la condición (interna).** Al juez no se le informa la condición, y la lista de palabras cambiada antes del juicio no se pensó para estas dos condiciones (Sección 4.5). Un juez podría inferir la condición por el estilo de una tabla. El estudio no tiene una medida de eso. Dos jueces de proveedores distintos coinciden en el 93,2% de las decisiones (Sección 5.3), lo que delimita el ruido del juez y no un sesgo que puedan compartir. El juez decide además todos los riesgos de referencia de una respuesta en una sola llamada, de modo que esas decisiones no son independientes (Flores, 2026b).

**9. Los repositorios son públicos y pueden estar en los datos de entrenamiento de los modelos (externa).** Varias tareas usan proyectos de ejemplo ampliamente copiados (Tabla 2). El agente puede recordar trampas conocidas de esos proyectos, y la frase puede ayudar más a la recuperación que al razonamiento. Las dos condiciones usan los mismos repositorios, de modo que esto afecta

al significado del efecto y no a su existencia; valen las cautelas sobre fuga de datos en estudios con LLM (Sallou et al., 2024).

**10. Dos modelos de un proveedor son una prueba débil de generalidad (externa).** Claude Opus 5.5 y Claude Sonnet 5.5 comparten proveedor, *harness* y modo. En el estudio de Patel et al. (2026), un *prompt* de predicción movió a dos modelos de una misma familia en sentidos opuestos, de modo que la concordancia entre los dos modelos aquí es informativa; nada dice sobre modelos de otro proveedor, otro agente, bases de código privadas o una sesión interactiva.

**11. La frase es medio pre-mortem (constructo).** El ejercicio de Klein incluye la suposición de que el plan fracasó (Klein, 2007). La frase la omite, y la condición con el marco completo de Klein, que el protocolo permitía como exploratoria, no se ejecutó (Sección 8). Este artículo usa "pre-mortem" para la familia de la instrucción y no hace ninguna afirmación sobre la suposición de fracaso.

**12. Tiempo y autor (interna).** La recolección se pausó durante cuatro días y se reanudó en la misma versión de Claude Code; como el orden fue una permutación con semilla, las dos condiciones se reparten entre los dos períodos (Sección 8). El autor escogió las tareas conociendo el resultado anterior, las fijó antes de que existieran las listas de referencia (Tabla 4) y mantiene ASP, la herramienta cuyo protocolo esta frase superó (Flores, 2026a).

**13. En uno de los modelos, las condiciones difieren en una llamada a una herramienta (interna).** En Opus 5.5, la solicitud simple llevó al agente a guardar un archivo de plan en 59 de 60 ejecuciones, frente a 22 de 60 con la frase (Sección 5.6). Un agente que escribió el plan en un archivo puede dar una respuesta final más corta, y la tabla se puntúa a partir de la respuesta final. Sonnet 5.5 es el control: no guardó archivo de plan en ninguna ejecución, y su mejora es la mayor de las dos, 12,4 puntos (Tabla 5). Entre las ejecuciones de Opus 5.5 que guardaron el archivo, la

diferencia es de 7,5 puntos en los 10 estratos en que las dos condiciones permanecen (Flores, 2026b). Guardar el archivo es también una respuesta a la solicitud, de modo que estas comparaciones describen y no aíslan una causa. Muestran que la mejora aparece sin la llamada a la herramienta, en Sonnet 5.5; la llamada puede explicar parte del menor costo en Opus 5.5.

**14. Tres límites señalados en la revisión (constructo).** Con la frase el agente usó menos turnos (Tabla 8), de modo que pudo haber leído menos del repositorio antes de responder, y este estudio no mide cuánto se apoyan en el código los riesgos enumerados. La tabla tiene un tope de diez filas y las dos condiciones casi lo llenan, de modo que la medida es qué riesgos entran en los diez primeros, y no cuántos riesgos sabría nombrar el agente. El juez decide todos los riesgos de referencia de una respuesta en una sola llamada y en orden fijo, de modo que no se excluye un efecto de posición sobre sus decisiones (Flores, 2026b).

**Validez de la conclusión.** Una hipótesis principal carga con la decisión, mediante una regla conjunta: la prueba combinada y la prueba dentro de cada modelo deben quedar todas por debajo de ,05 (Sección 4.6); las medidas secundarias se reportan como tales, y todo análisis no planificado se etiqueta como no planificado (Sección 8), la separación que el prerregistro existe para proteger (Nosek et al., 2018). Cinco réplicas por celda dan medias de estrato toscas; la prueba combina 24 estratos y no depende de ninguno de ellos (Tabla 6). Las pruebas tratan las doce tareas como fijas, y los dos modelos las comparten; remuestrear tareas enteras da de 8,2 a 14,5 puntos para la diferencia prerregistrada y de 2,2 a 11,1 para la segunda vara de medir (Sección 5.8). Todo valor de p y todo intervalo aquí es condicional a las listas de referencia, a su fusión y a los veredictos tal como se recogieron; ninguno incluye la variación que añadiría una nueva petición de listas, una nueva fusión o un nuevo juicio, y la distancia entre las dos varas de medir y entre los dos jueces de la segunda muestra que esa variación no es pequeña.

## 8. Desviaciones del protocolo

El archivo del protocolo registra cada desviación con su fecha (Flores, 2026b). En resumen:

1. **Un conflicto de carpeta corregido en los primeros minutos.** Las ejecuciones numeradas t001 a t090 fallaron al clonar, antes de toda llamada al modelo, porque la carpeta de las copias de los repositorios aún guardaba copias del estudio anterior. Las t091 a t096, cuyas carpetas no existían, corrieron antes y se mantuvieron. Se renombró la carpeta y la recolección se reanudó en el orden con semilla. *Prompts*, modelos, tareas, puntuación y análisis no cambiaron.
2. **Una pausa de cuatro días.** La recolección se detuvo tras 38 ejecuciones, al apagarse el ordenador que la corría. Las ejecuciones en curso en ese momento no dejaron registro y se corrieron de nuevo al reanudar. Su número y sus condiciones no se registraron, de modo que una selección por duración entre las ejecuciones en curso en ese momento no puede medirse ni descartarse. El ordenador actualizó Claude Code durante la pausa; las ejecuciones restantes corrieron en la versión de las primeras 38, instalada por separado.
3. **Rutas de máquina eliminadas de tres respuestas.** Las t006, t022 y t023 citaron un archivo del ordenador del operador por su ruta completa, fuera de la tabla de riesgos. La ruta se cambió por un marcador, cada archivo corregido recibió nueva entrada en el registro, y, desde la reanudación, el ejecutor aplica los mismos marcadores antes de guardar.
4. **Tres ejecuciones con error de red,** tratadas como se fijó de antemano y examinadas en la Sección 5.5.

5. **Una pausa en la puntuación.** El juez principal alcanzó su límite de uso tras 225 de 240 veredictos. Los 15 restantes los produjo el mismo modelo, con el mismo comando, al renovarse el límite. No se sustituyó a ningún juez.

6. **Una condición exploratoria que no se ejecutó.** El protocolo permitía, si sobraba uso, una tercera condición con el marco completo de Klein: “Imagine it is a month from now and this change has failed. Before planning, list why it failed.” (en español: “Imagina que pasó un mes y este cambio fracasó. Antes de planificar, enumera por qué fracasó.”) No tiene ejecuciones y no guarda relación con el resultado.

7. **Análisis no planificados,** cada uno etiquetado donde aparece: el análisis de sensibilidad y su límite opuesto, el intervalo *bootstrap*, el κ de Cohen, la comparación de tablas completas, la comparación del archivo de plan y la descripción por elemento.

8. **La segunda vara de medir,** exploratoria y especificada después del resultado (Sección 5.8). El juez principal alcanzó su límite tras 208 de sus 240 veredictos, que se reanudaron tras la renovación con el mismo modelo y comando. El segundo juez lo alcanzó tras 11 veredictos, y no se había renovado al escribirse esta versión. Después se le añadieron tres comprobaciones, pedidas por un revisor (Sección 5.8).

9. **Una repetición de las tres ejecuciones con error y un tercer juez,** ambos no planificados y fijados en un segundo anexo antes de cualquier llamada (Secciones 5.5 y 5.8). El juez principal alcanzó su límite de uso en la línea de comandos tras 4 de sus 9 veredictos sobre las repeticiones. Por decisión del autor, registrada en un tercer anexo antes de las llamadas, los 9 los produjo el mismo modelo por una API de pago, con una comprobación de canal en 30 ejecuciones selladas. Ese anexo decía que la respuesta completa de la API se guardaría con cada veredicto; el código guardó el texto de la

respuesta, el modelo, el proveedor, el recuento de tokens y el identificador de la llamada. Los veredictos de la línea de comandos siguen pendientes.

## 9. Conclusión

Una frase que abre una solicitud de planificación en lugar de “Plan this change.”, “Before planning, list the ways this change could go wrong.”, elevó de 46% a 58% la parte de riesgos enumerados de forma independiente que cubre el plan de un agente de código, en un estudio confirmatorio prerregistrado de 240 ejecuciones, en los dos modelos probados y según dos jueces (Tabla 5). Los riesgos enumerados no se juzgaron menos concretos, y no se observó costo adicional (Sección 5.4; Tabla 8). El hallazgo replica un resultado secundario de un estudio anterior en tareas nuevas y en un segundo modelo (Flores, 2026a).

Los límites son los de la Sección 7: la vara de medir y los jueces eran modelos, la vara se obtuvo con palabras próximas a las de la frase, y una segunda vara, obtenida con otras palabras, da una ganancia menor, de 6,7 puntos (Sección 5.8), los dos agentes comparten proveedor, y el desenlace es lo que dice la tabla de un plan, no lo que hace el código. Los próximos pasos salen de ellos, en el orden previsto: (i) un ingeniero humano que puntúe una muestra de las listas de referencia y las decisiones de los jueces, y riesgos de referencia tomados de defectos reales, que ningún *prompt* provocó; (ii) el mismo diseño en un modelo de otro proveedor; (iii) una ablación que separe listar los riesgos *antes de planificar* de listarlos sin más, y que añada la suposición de fracaso de Klein (Klein, 2007); y (iv) un desenlace en código: implementar los planes de las dos condiciones y ejecutar las pruebas de los proyectos, como hacen Patel et al. (2026) con un *prompt* más largo.

## 10. Disponibilidad de datos y código

El material detrás de cada número de este artículo está en el repositorio público anticipating-shadow-points (Flores, 2026b), en la carpeta benchmark/study-2026-11b de la rama v2, a partir del *commit* d880cf0; la *release* 2.0.0, la versión citada, es anterior a este estudio. La Tabla 11 describe la carpeta.

Un único comando, python run_all.py, recalcula cada resultado del estudio reportado en este artículo a partir de las respuestas y de los veredictos guardados y no llama a ningún modelo; exige Python 3.11 o más reciente (Flores, 2026b). Volver a correr las ejecuciones exige una suscripción a Claude y acceso a los dos modelos jueces, que pueden haber cambiado desde noviembre de 2026.

**Tabla 11**
*Lo que el repositorio contiene para este artículo*

| **Material** | **Dónde en el repositorio** | **Qué permite comprobar** |
|---|---|---|
| Protocolo, desviaciones y los tres anexos | PROTOCOL.md, yardstick2/, addendum2/, addendum3/ | Qué se fijó antes de los datos y qué cambió después |
| Tareas y *prompts* | tasks.json, run.py, prompts/ | El texto exacto que recibió cada ejecución y revisor |
| Listas de referencia y su fusión | lists/ | Los 369 elementos de revisores y los 100 riesgos de referencia |
| Las 240 respuestas y registros | runs/ | Cada respuesta y su registro |
| Los 720 veredictos | judge/ | Cada decisión, con su *prompt* |
| Análisis y verificaciones | analyze.py, sensitivity.py, describe.py; tests/test_paper3_numbers.py | Cómo se calcula cada número, y que el texto concuerda con los datos |
| Registro y sellos de tiempo | ledger.jsonl, *.ots | Cada archivo coincide con su registro más reciente, y el orden de los eventos sellados (límites: Sección 4.7) |
| Revisión externa del manuscrito | docs/paper/review/ | Los informes detrás de la Sección 7 |
| El estudio de origen | benchmark/study-2026-11/ | El mismo material para las 90 ejecuciones de la Sección 3 |

*Nota.* Las rutas sin carpeta delante están dentro de benchmark/study-2026-11b/ (Flores, 2026b). A los registros de ejecución se les quitaron las rutas de máquina.

## 11. Declaraciones

**Conflicto de intereses.** El autor mantiene ASP, el *plugin* cuyo protocolo el estudio de origen comparó con la frase probada, y dirige el laboratorio que lo publica. El resultado reportado aquí es desfavorable a ese protocolo (Flores, 2026a).

**Financiación.** Este trabajo no recibió financiación externa. El autor pagó el uso de los modelos.

**Contribución del autor.** Autor único: concepción, diseño, recogida de datos, análisis, redacción y revisión.

**Uso de herramientas de IA.** Los modelos de lenguaje tienen cuatro papeles en este estudio, todos descritos en la Sección 4: los agentes probados, los redactores de las listas de referencia, los jueces y los revisores ciegos del manuscrito (Flores, 2026b). Un asistente de IA se usó también en la programación, en el análisis y en la redacción y traducción del texto, bajo la dirección del autor. El autor comprobó cada número con los datos y responde por el contenido.

**Referencias**

Anthropic. (2026). *Choose a permission mode* [Software documentation]. https://code.claude.com/docs/en/permission-modes

Baltes, S., Angermeir, F., Arora, C., Muñoz Barón, M., Chen, C., Böhme, L., Calefato, F., Ernst, N., Falessi, D., Fitzgerald, B., Fucci, D., He, J., Treude, C., Kalinowski, M., Lambiase, S., Russo, D., Lungu, M., Martinez Montes, C., Prechelt, L., Ralph, P., van Tonder, R., & Wagner, S. (2027). Guidelines for empirical studies in software engineering involving large language models. *Empirical Software Engineering, 32*(2), Article 64. https://doi.org/10.1007/s10664-026-10922-3

Caraeni, A., Shick, A., & Lan, A. (2026). *Cognitive offloading in agile teams: How artificial intelligence reshapes risk assessment and planning quality*. arXiv:2604.13814. https://arxiv.org/abs/2604.13814

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., & Stoica, I. (2025). Why do multi-agent LLM systems fail? *Advances in Neural Information Processing Systems, 38*, 135676–135729. https://doi.org/10.52202/085713-4082

Cheng, Z., & Mastropaolo, A. (2026). *An empirical study on the effects of system prompts in instruction-tuned models for code generation*. arXiv:2602.15228. https://arxiv.org/abs/2602.15228

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*(1), 37–46. https://doi.org/10.1177/001316446002000104

Ernst, M. D. (2004). Permutation methods: A basis for exact inference. *Statistical Science, 19*(4), 676–685. https://doi.org/10.1214/088342304000000396

Flores, C. U. (2026a). *ASP: An operational pre-mortem skill for LLM coding agents — An experience report* (Version 2.4) [Technical report]. Zenodo. https://doi.org/10.5281/zenodo.24065968

Flores, C. U. (2026b). *ASP — Anticipating Shadow Points: Claude Code plugin and replication package* (Version 2.0.0; branch v2, commit d880cf0) [Computer software]. GitHub. https://github.com/ulissesflores/anticipating-shadow-points https://doi.org/10.5281/zenodo.20276631

Fröhling, L., Giaconia, A., Bogucka, E. P., & Quercia, D. (2026). Agent-supported foresight for AI systemic risks: AI agents for breadth, experts for judgment. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems* (pp. 1–48). ACM. https://doi.org/10.1145/3772318.3790712

International Electrotechnical Commission. (2019). *Risk management — Risk assessment techniques* (IEC 31010:2019, Edition 2.0). https://webstore.iec.ch/en/publication/59809

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2024). *SWE-bench: Can language models resolve real-world GitHub issues?* ICLR 2024. https://arxiv.org/abs/2310.06770

Kharma, M., Sabbah, A., Alkhanafseh, M., Hammoudeh, M., & Mohaisen, D. (2026). *An empirical evaluation of LLM-generated code security across prompting methods*. arXiv:2605.24298. https://arxiv.org/abs/2605.24298

Kim, Y., Kim, T., Park, E., Park, C., Breazeal, C., McDuff, D., & Park, H. W. (2026). *InvThink: Premortem reasoning for safer language models* (Version 3). arXiv:2510.01569. https://arxiv.org/abs/2510.01569

Klein, G. (2007). Performing a project premortem. *Harvard Business Review, 85*(9), 18–19. https://hbr.org/2007/09/performing-a-project-premortem

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems, 35*. https://arxiv.org/abs/2205.11916

Mitchell, D. J., Russo, J. E., & Pennington, N. (1989). Back to the future: Temporal perspective in the explanation of events. *Journal of Behavioral Decision Making, 2*(1), 25–38. https://doi.org/10.1002/bdm.3960020103

Nosek, B. A., Ebersole, C. R., DeHaven, A. C., & Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences, 115*(11), 2600–2606. https://doi.org/10.1073/pnas.1708274114

Panickssery, A., Bowman, S. R., & Feng, S. (2024). LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems, 37*. https://doi.org/10.52202/079017-2197

Patel, S., Liao, M., Henderson, P., He, H., & Chen, V. (2026). *Do coding agents catch their own bugs? (They don't.)*. arXiv:2611.01181. https://arxiv.org/abs/2611.01181

Phipson, B., & Smyth, G. K. (2010). Permutation p-values should never be zero: Calculating exact p-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology, 9*(1), Article 39. https://doi.org/10.2202/1544-6115.1585

Sallou, J., Durieux, T., & Panichella, A. (2024). Breaking the silence: The threats of using LLMs in software engineering. In *Proceedings of the 2024 ACM/IEEE 44th International*

*Conference on Software Engineering: New Ideas and Emerging Results* (pp. 102–106). ACM. https://doi.org/10.1145/3639476.3639764

Schoenegger, P., Jones, C. R., Tetlock, P. E., & Mellers, B. (2025). *Prompt engineering large language models' forecasting capabilities*. arXiv:2506.01578. https://arxiv.org/abs/2506.01578

Sclar, M., Choi, Y., Tsvetkov, Y., & Suhr, A. (2024). *Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting*. ICLR 2024. https://arxiv.org/abs/2310.11324

Tony, C., Díaz Ferreyra, N. E., Mutas, M., Dhif, S., & Scandariato, R. (2025). Prompting techniques for secure code generation: A systematic investigation. *ACM Transactions on Software Engineering and Methodology, 34*(8), Article 225. https://doi.org/10.1145/3722108

Veinott, B., Klein, G. A., & Wiggins, S. (2010). Evaluating the effectiveness of the PreMortem technique on plan confidence. In *Proceedings of the 7th International ISCRAM Conference*. https://web.archive.org/web/20200928015943/http://idl.iscram.org/files/veinott/2010/1049_Veinott_etal2010.pdf

Veinott, E. S., & Lehman, B. R. (2024). Adaptive planning: Comparing human and AI responses in premortem planning. In *HCI International 2024 – Late breaking papers* (pp. 256–268). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-76827-9_15

Wang, H., Cao, Y., Lin, L., & Chen, J. (2026). *PreFlect: From retrospective to prospective reflection in large language model agents*. arXiv:2602.07187. https://arxiv.org/abs/2602.07187

Wang, H., Li, T., Deng, Z., Roth, D., & Li, Y. (2024). Devil's advocate: Anticipatory reflection for LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024* (pp. 966–978). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.findings-emnlp.53

Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., & Lim, E.-P. (2023). Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 2609–2634). https://doi.org/10.18653/v1/2023.acl-long.147

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer. https://doi.org/10.1007/978-3-642-29044-2

Wu, J. (2026). *AssumptionMiner: Extracting, tracing, and revising implicit assumptions in LLM code generation*. arXiv:2607.22898. https://arxiv.org/abs/2607.22898

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. NeurIPS 2023 D&B. https://arxiv.org/abs/2306.05685

## Apéndice A — Las solicitudes exactas

La Tabla A1 da las peticiones principales del estudio confirmatorio; las peticiones que fusionan las listas y juzgan la relevancia, y la segunda petición a los revisores, están en los archivos que enumera la Tabla 11, y cada cliente añade instrucciones propias que no se muestran. Las llaves marcan las únicas partes que cambian de una llamada a otra. Los archivos que guardan esos textos están enumerados en la Tabla 11 (Flores, 2026b).

**Tabla A1**

*Las solicitudes enviadas al agente, a los revisores y al juez*

| Quién recibe | Texto |
|---|---|
| El agente, condición simple | {change request} Plan this change. {output contract} |
| El agente, condición con la frase | {change request} Before planning, list the ways this change could go wrong. {output contract} |
| El contrato de salida, en las dos condiciones | Non-interactive run: state your assumptions instead of asking. Do not edit any file. Answer in English. End your answer with the main risks of this change as one markdown table with columns #, Risk, Why it bites here, Mitigation (at most 10 rows, short cells), then one line starting “Riskiest:” naming the riskiest one. |
| Cada revisor | You are a senior software engineer reviewing a change request before anyone implements it. The repository in the current directory is the codebase. Read whatever files you need. Do not modify any file. Change request: {change request} List the concrete ways this change could go wrong in this codebase if it were implemented without care: the problems a careful senior engineer would want addressed before starting. Each item must be specific to this repository and this request, one or two sentences. Answer with a numbered list of 6 to 12 items and nothing else. |
| El juez de la cobertura | A plan for a software change ends with a table of risks. Decide, for each reference problem, whether any row of the table addresses the same underlying problem (names it as a risk or mitigates it specifically), even if worded differently. Change request: {change request} Reference problems: {numbered reference risks} Rows of the risk table: {rows} Answer directly from the text above; do not run commands or use tools. Answer with JSON only, one key per reference number. |

*Nota.* Los saltos de línea se eliminaron. La última frase del juez va seguida, en el original, de un ejemplo del JSON esperado.

## Apéndice B — Una lista de referencia completa

La Tabla B1 muestra la lista de referencia de la tarea U08, la más corta de las doce, con la parte de las ejecuciones de cada condición cuya tabla cubre cada riesgo según el juez principal. Ilustra el patrón de la Sección 5.7: riesgos que todo plan cita en las dos condiciones (filas 1 y 5), riesgos que la frase vuelve regulares (filas 4 y 6) y un riesgo que las dos condiciones casi siempre pasan por alto (fila 2). Las listas completas están en el repositorio (Flores, 2026b).

**Tabla B1**

*Los seis riesgos de referencia de la tarea U08 (cambiar el digest de firma por defecto de SHA-1 a SHA-256) y su cobertura*

| # | Riesgo de referencia, abreviado | Revisores que lo citaron | Simple | Frase |
|---|---|---|---|---|
| 1 | Los tokens firmados por versiones anteriores con la configuración por defecto fallan en la validación tras la actualización, invalidando sesiones en proyectos dependientes como Flask. | 3 | 100% | 100% |
| 2 | Cambiar el valor por defecto en el *signer* y no en la clase del algoritmo HMAC deja los dos valores por defecto desincronizados. | 3 | 10% | 20% |
| 3 | La derivación de clave también usa el *digest*, de modo que la clave derivada cambia de valor y pasa de 20 a 32 bytes. | 3 | 40% | 50% |
| 4 | Las pruebas afirman salidas que coinciden con firmas SHA-1. | 2 | 70% | 100% |
| 5 | Las firmas pasan de 27 a 43 caracteres y pueden desbordar columnas de base de datos, límites de URL o tamaños de *cookie*. | 2 | 100% | 100% |
| 6 | Los ejemplos de documentación y los *doctests* guardan salidas SHA-1 y afirman que SHA-1 es el valor por defecto. | 2 | 40% | 90% |

*Nota.* Diez ejecuciones por condición (cinco por modelo). Los seis riesgos llevan la redacción del revisor Gemini. Aquí, token es un valor firmado; el *digest* es la función hash usada en la firma; el *signer* es la clase que firma; HMAC es un método estándar de firmar con una clave secreta y una función hash; los *doctests* son ejemplos de la documentación que se ejecutan como pruebas.

## Apéndice C — Los textos en inglés, traducidos

Todo texto que un modelo recibió en este estudio estaba en inglés, y el artículo lo reproduce tal como se envió. Este apéndice traduce esos textos para quien lee en español: la Tabla C1 trae las doce solicitudes de cambio de la Tabla 2, y la Tabla C2, las solicitudes de la Tabla A1. Las traducciones son del autor y no se usaron en ninguna ejecución (Flores, 2026b).

**Tabla C1**
*Las doce solicitudes de cambio, traducidas*

| Tarea | Solicitud de cambio, en español |
|---|---|
| U01 | Exigir que los usuarios nuevos confirmen su dirección de correo electrónico antes de poder iniciar sesión. |
| U02 | Permitir que los autores cambien el título de un artículo, y con él el *slug* de la URL, sin romper los enlaces ya compartidos. |
| U03 | Permitir que los moderadores oculten comentarios abusivos y que el autor del comentario apele en un plazo de 7 días. |
| U04 | Bloquear a los usuarios por la combinación de nombre de usuario y dirección IP, y no solo por la dirección IP, como nuevo valor por defecto. |
| U05 | Hacer que las sesiones se renueven por defecto, de modo que cada solicitud extienda la validez de la sesión. |
| U06 | Permitir que cada usuario descargue todos sus datos (perfil, publicaciones y comentarios) en un único archivo. |
| U07 | Sustituir por UUID los identificadores enteros que expone la API, manteniendo en funcionamiento a los clientes actuales de la API. |
| U08 | Cambiar el algoritmo de firma por defecto de SHA-1 a SHA-256. |
| U09 | Guardar las contraseñas con un esquema de hash más fuerte, sin obligar a los usuarios actuales a restablecer su contraseña. |
| U10 | Hacer que cada lista de tareas y cada elemento pertenezcan a una organización (*tenant*), de modo que cada empresa vea solo sus propios datos. |
| U11 | Permitir que los dueños de mascotas reserven citas por internet, sin que se reserven dos citas para el mismo veterinario a la misma hora. |
| U12 | Hacer que la *cookie* del *middleware* de sesión sea Secure y SameSite=Strict por defecto. |

**Tabla C2**

*Las solicitudes enviadas al agente, a los revisores y al juez, traducidas*

| Quién recibe | Texto, en español |
|---|---|
| El agente, condición simple | {solicitud de cambio} Planifica este cambio. {contrato de salida} |
| El agente, condición con la frase | {solicitud de cambio} Antes de planificar, enumera las formas en que este cambio puede salir mal. {contrato de salida} |
| El contrato de salida, en las dos condiciones | Ejecución no interactiva: declara tus supuestos en lugar de preguntar. No edites ningún archivo. Responde en inglés. Termina la respuesta con los principales riesgos de este cambio en una tabla markdown con las columnas #, Riesgo, Por qué pesa aquí y Mitigación (como máximo 10 filas, celdas cortas) y, después, una línea que empiece por "Riskiest:" y señale el más arriesgado. |
| Cada revisor | Eres un ingeniero de software sénior que revisa una solicitud de cambio antes de que alguien la implemente. El repositorio de la carpeta actual es la base de código. Lee los archivos que necesites. No modifiques ningún archivo. Solicitud de cambio: {solicitud de cambio} Enumera las formas concretas en que este cambio podría salir mal en esta base de código si se implementara sin cuidado: los problemas que un ingeniero sénior cuidadoso querría ver tratados antes de empezar. Cada elemento debe ser específico de este repositorio y de esta solicitud, de una o dos frases. Responde con una lista numerada de 6 a 12 elementos y nada más. |
| El juez de la cobertura | Un plan de cambio de software termina con una tabla de riesgos. Decide, para cada problema de referencia, si alguna fila de la tabla trata el mismo problema de fondo (lo nombra como riesgo o lo mitiga específicamente), aunque esté redactado de otro modo. Solicitud de cambio: {solicitud de cambio} Problemas de referencia: {riesgos de referencia numerados} Filas de la tabla de riesgos: {filas} Responde directamente a partir del texto anterior; no ejecutes comandos ni uses herramientas. Responde solo con JSON, una clave por número de referencia. |

*Nota.* Los textos originales, en inglés, están en la Tabla A1.

**Nota Técnica**

---

**Versión** 1.0 · noviembre de 2026

**Serie** Codex Hash Research Report 2026-03

**Licencia** Texto, figuras y datos: CC BY 4.0. Software complementario: MIT.

**El proyecto ASP.** ASP (Anticipating Shadow Points) es un *plugin* de código abierto para Claude Code que hace que el agente enumere lo que podría salir mal antes de escribir código. El *plugin* y el material de replicación están en un único repositorio público (Flores, 2026b). Un *whitepaper* describe cómo funciona el *plugin* y reporta la medición inicial de si mejora los planes de un agente, que no halló beneficio medible frente a una solicitud de una frase (Flores, 2026a). Este artículo es el estudio confirmatorio de esa frase que anunció el *whitepaper*.

**DOI.** Un DOI (Digital Object Identifier) es un enlace permanente a una obra depositada. Zenodo, archivo público de investigación mantenido por el CERN, guarda este artículo como registro propio; esta versión, 1.0, es https://doi.org/10.5281/zenodo.24410437. El *plugin*, los datos y el código forman un registro separado (Flores, 2026b), cuyo DOI de concepto, https://doi.org/10.5281/zenodo.20276631, abre siempre la *release* más reciente.

**Replicación.** Todo número que este artículo calcula a partir de sus datos puede recalcularse desde el repositorio sin llamar a un modelo; la Sección 10, Disponibilidad de datos y código, enumera lo que hay allí y cómo ejecutarlo.

**Sobre el autor**

**Carlos Ulisses Flores** es CTO e Investigador Jefe de Codex Hash Research Laboratory, un laboratorio independiente de I+D en São Paulo, Brasil. Construye sistemas de producción donde el fallo silencioso sale caro: históricamente en trading algorítmico, IoT embebido y criptografía en el borde, y hoy en herramientas de IA agéntica. Este artículo nació de medir el propio *plugin* de planificación para agentes de código y descubrir que una frase funcionaba mejor.

**Identificadores** · ORCID: 0000-0002-6034-7765 · Lattes (registro nacional de currículos de investigadores de Brasil): 6905246706890561

**Contacto** · c.ulisses@gmail.com · https://ulissesflores.com