

El contrato del residuo: la capa de auditoría que le falta a la revisión con IA

Las plataformas ya hacen que un modelo revise el trabajo de otro, de familia distinta. Lo que ninguna registra es lo que esa revisión deja abierto.

Nicolás Rocchia · Pelatech · Agosto de 2026 · pelatech.com.ar/blog/contrato-del-residuo

EL HUECO

La ronda ya es **commodity**. El registro no existe.

GitHub incorporó a Copilot CLI una segunda opinión entre familias de modelos, disponible en general desde junio de 2026. La premisa es correcta. Lo que produce muerte con la sesión.

Cuando un modelo escribe y otro critica, la parte más valiosa de esa crítica es justamente lo que **no** cerró: el supuesto sin verificar, el intercambio que alguien aceptó, la escalación que espera una decisión humana. Ese es el residuo de la revisión.

Hoy ese residuo no tiene registro, no tiene dueño y no lo hace cumplir nadie, en ninguna plataforma. La crítica alimenta el razonamiento del agente y desaparece: quién revisó a quién, con qué independencia, qué se descartó sin evidencia. En el propio repositorio de Copilot CLI hay reportes de usuarios extrayendo evidencia a mano de los logs de proceso para probar qué modelo revisó a cuál, porque la garantía de independencia puede fallar en silencio y no queda asentada en ningún lado.

Este caso documenta cómo atacué ese hueco: primero el método con datos, después la herramienta, después las afirmaciones públicas. Todo fechado y verificable, porque de eso se trata.

LA CONSTRUCCIÓN

Método, herramienta, prueba y reclamo. **En ese orden.**

El orden importa: cada pieza existe porque la anterior la respalda, y las fechas públicas lo muestran.

Primero, el método

Operé 52 días de revisión adversarial disciplinada sobre cinco proyectos profesionales: un asistente genera, otro de familia distinta ataca, y cada hallazgo llega a un estado terminal. Publiqué el método con los datos antes de escribir una línea de la herramienta: 91 eventos de revisión documentados, con los falsos positivos y lo que el método no detectó incluidos. Es el informe de experiencia que ya vive en este blog, con DOI de Zenodo.

Después, la herramienta

disensor (MIT, en PyPI y en el Marketplace de GitHub) corre la ronda y la deja declarada. Arma el paquete que recibe el revisor, ejecuta el asistente que haya en la máquina, mide el árbol de trabajo antes y después, y ancla el resultado a los commits que efectivamente se revisaron. De ahí sale la declaración JSON: quién generó, quién atacó (familia de modelo más el hash de la consigna adversarial), cada hallazgo con su estado terminal, y el residuo declarado. Un validador determinístico aplica las reglas del artefacto y una acción de CI aplica los chequeos de compuerta sobre cada PR. En el camino de validación no hay ningún modelo de lenguaje ni hacen falta claves de API: el control solo lee artefactos que viven en el repositorio.

El revisor no está atado a ningún proveedor: sirve cualquier asistente con línea de comandos, y no hay que copiar y pegar nada entre un modelo y el otro. Cuando no hay un segundo modelo de otra familia disponible, la ronda corre igual y queda declarada como el modo degradado que es, con la correlación que no se cubrió asentada como residuo. Una ronda degradada declarada vale más que una ronda que no ocurrió.

La prueba, en el propio repositorio

El repositorio de disensor se gobierna a sí mismo: cada PR pasa por la versión publicada de disensor, y el directorio de declaraciones guarda el registro real y fechado de su propio desarrollo, incluidas las rondas que rechazaron mis propios cambios. Las releases se publican por Trusted Publishing con OIDC: cero tokens en cualquier máquina.

El reclamo público, fechado

Presenté el primer pedido registrado de que las críticas de la revisión dejen un registro verificable, en el propio tracker de Copilot CLI, motivado por los reportes de independencia

fallando en silencio. Si la plataforma construye esa capa mañana, el registro público muestra quién la especificó primero y dónde la leyeron.

LOS NÚMEROS

Todo **verificable**, nada redondo.

Cada cifra tiene una fuente pública: el informe con DOI, el repositorio, los registros de PyPI.

91 Eventos de revisión medidos antes de publicar la herramienta	52 Días de piloto documentado sobre cinco proyectos
363 Tests, en un repositorio gateado por su propia herramienta	03 Familias de modelos orquestadas en las rondas reales

INGENIERÍA

Decisiones que puedo **defender** en cualquier sala.

Un control que decide si un merge se permite no puede tener buena voluntad como mecanismo. Estas son las decisiones que lo sostienen.

- Fail-closed en todas partes.

Sin rango de PR no hay aprobación; una política malformada rechaza en vez de aplicar defaults. Un control que se degrada con gracia bajo una condición que el atacante controla no es un control.
- La política se lee de la rama destino.

Un PR no puede relajar las reglas que lo juzgan, porque las reglas vigentes son las de la rama a la que quiere entrar.
- La evidencia es de solo agregar.

La historia de revisión no puede ser reescrita por el cambio que revisó.
- El pin es un commit, y lo resuelve un comando.

Un tag se puede mover, así que no es raíz de confianza. Después de un incidente real (un SHA tipeado de memoria que mi propia suite de tests detuvo), la resolución dejó de ser tarea manual: la hace la herramienta, esquivando la trampa del tag anotado que devuelve el objeto tag en vez del commit.

La familia de defectos más instructiva del piloto

: el test que verifica el sustituto en vez del comportamiento. La documenté nueve veces, cada una cazada por el revisor de la otra familia y asentada en una declaración pública. Un mock que aceptaba cualquier argumento, un test que leía el código fuente en vez de ejecutarlo, una captura que escondía un crash real de encoding.

REGISTRO PÚBLICO

El escrutinio que el método **invita**.

Un método sobre honestidad de revisión se juzga por cómo responde cuando lo revisan a él.

La auditoría que no pedí

Al presentar la herramienta en una comunidad de spec-driven development, un revisor independiente leyó el código fuente antes de comentar, verificó que la frontera epistémica declarada existe en el código, y presionó exactamente donde había que presionar: evidencia vacua, separación de roles, el riesgo de que el control se vuelva ceremonia. Su apertura: que esto cierra un hueco real en todos los harnesses que revisó. Su cierre, después del intercambio completo: una mejora significativa sobre las alternativas. Ese hilo público es el tipo de escrutinio para el que la herramienta existe.

Las rondas, vividas

El desarrollo mismo corre sobre el método: modelos de Anthropic generan, modelos de OpenAI y de Google atacan. Hay un evento registrado que lleva dos revisores porque al primero se le agotó la cuota a mitad de la ronda; la declaración lo dice tal cual, porque las declaraciones registran lo que pasó, no lo que debería haber pasado.

Los límites, por escrito

La compuerta detecta el campo vacío y el marcador genérico, no la mentira bien escrita. El muestreo humano queda en el circuito por diseño, y los problemas abiertos son issues públicos, no letra chica: representar evidencia externa, y correlacionar el residuo declarado con los defectos que después escapan.

Este caso, para llevar.

El caso en PDF para adjuntar o imprimir, y las fuentes primarias para verificar cada afirmación.

ARCHIVO		PDF + FUENTES
ES	Descargar el caso contrato-del-residuo.pdf · 6 páginas · 301 kB · español	↓
EN	Download the English version residue-contract.pdf · 6 páginas · 297 kB · traducción del autor	↓
GIT	El repositorio de disensor MIT · el código, las declaraciones reales de su desarrollo y los issues abiertos	↗
WEB	disensor.dev La página del producto, con la guía de adopción	↗

CONTACTO

¿Tu equipo revisa código generado con IA y no puede responder qué dejó abierto la revisión?

Ese es el problema que vengo resolviendo, con el método y la herramienta a la vista. Si lo estás sufriendo, o si querés discutir el diseño, escribime.

EMAIL info@pelatech.com.ar

LINKEDIN nicolas-rocchia ↗

TAMBIÉN EN PELATECH		03
01	Desacuerdo controlado El informe de experiencia detrás de este caso, con DOI	→

02 Receta mínima de adopción



El método en una carilla, para aplicarlo

03 Proyectos



Productos propios y trabajo a medida para clientes