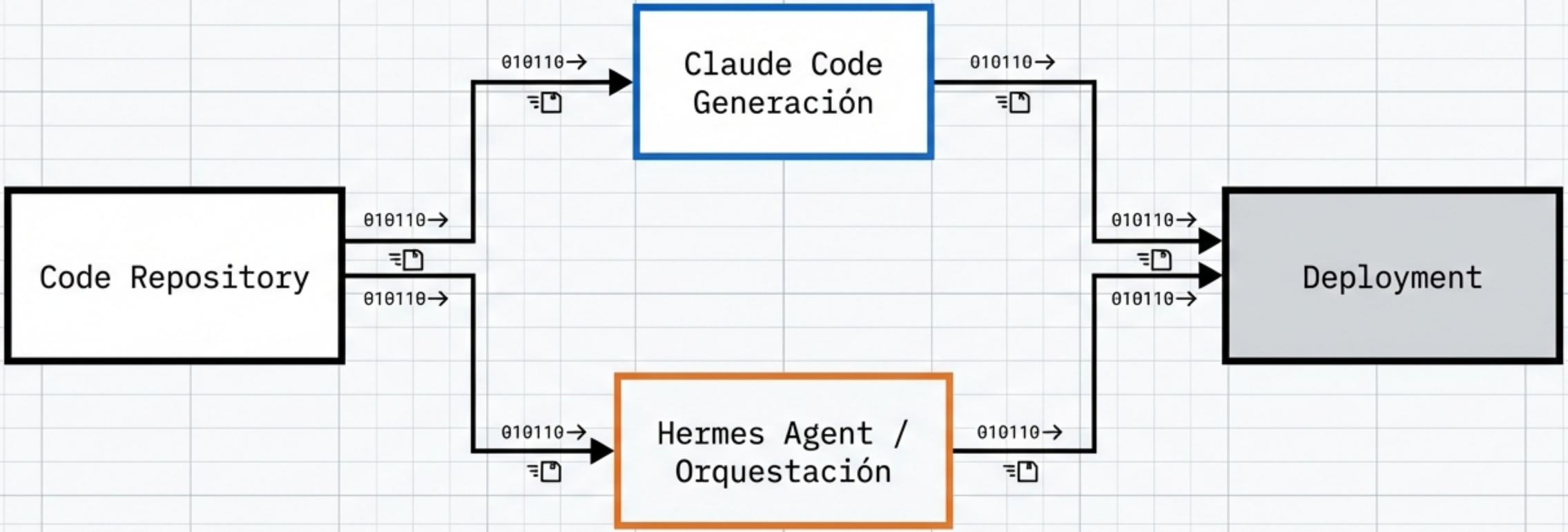
Arquitectura Multi-Agente para CI/CD: Automatización Autónoma con Hermes y Claude Code

Del Code Review al Deployment: Un enfoque basado en agentes para flujos de trabajo de ingeniería continuos.

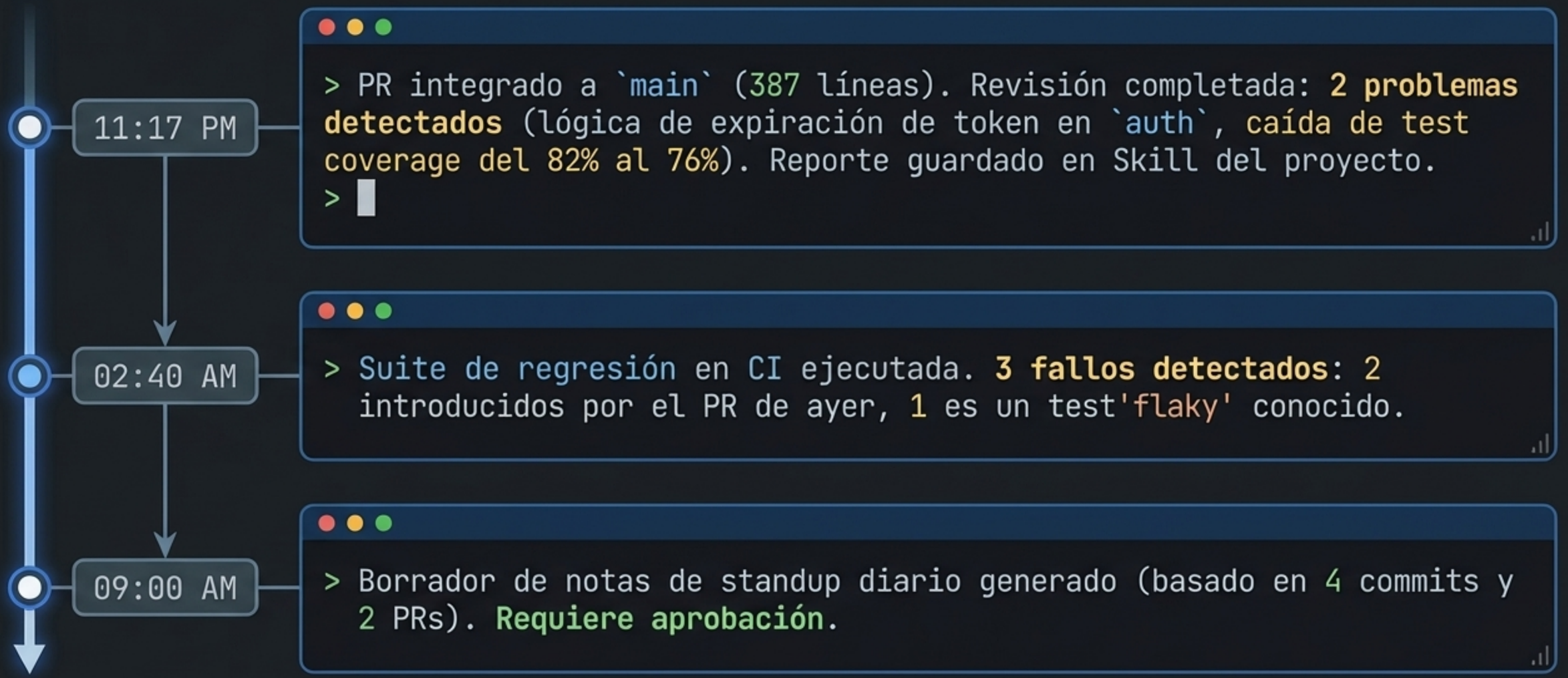


El Paradigma Multi-Agente: División del Trabajo

Dimensión	Claude Code (Generador)	Hermes Agent (Orquestador)
Modo de Interacción	Tiempo real, conversacional in-situ	Ejecución en background, reportes programados
Competencias (Strengths)	Escritura de código, refactorización, debugging	Monitoreo, auditoría, síntesis, calendarización
Horizonte Temporal	Sesión única	Continuo, abarca días y semanas
Mecanismo (Trigger)	Iniciado por el usuario	Basado en eventos o cron

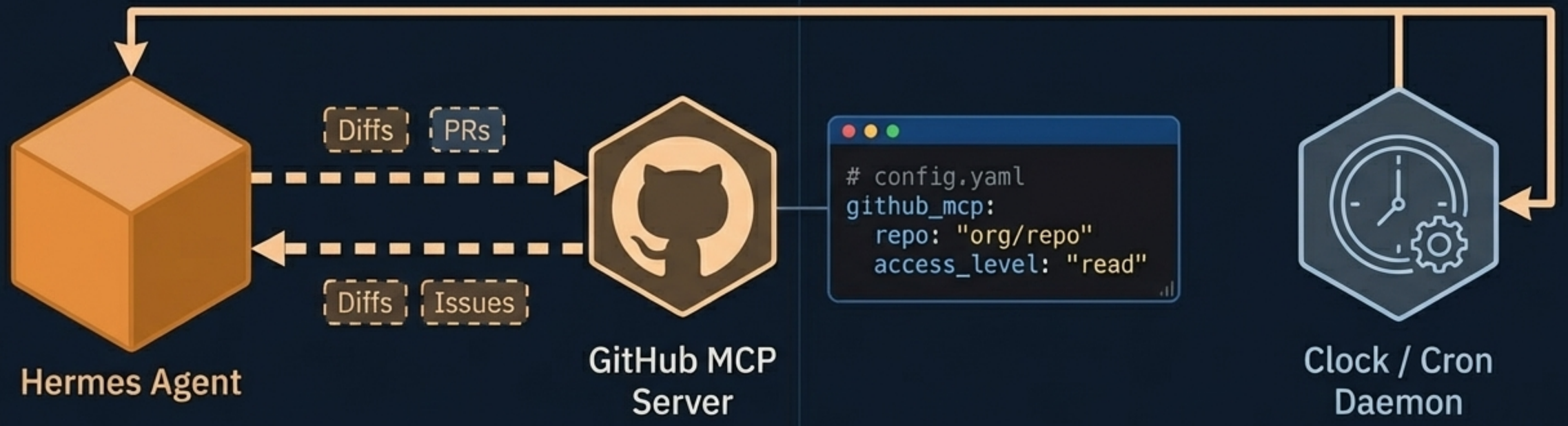
El Artesano y el Mayordomo: Claude Code construye la lógica; **Hermes Agent** asegura la integridad del sistema a nivel global.

Output del Sistema: Observabilidad y Ejecución Asíncrona



Infraestructura habilitadora: Cron Scheduling + GitHub MCP + Memoria de Estado.

Infraestructura Base: Model Context Protocol (MCP) y Cron

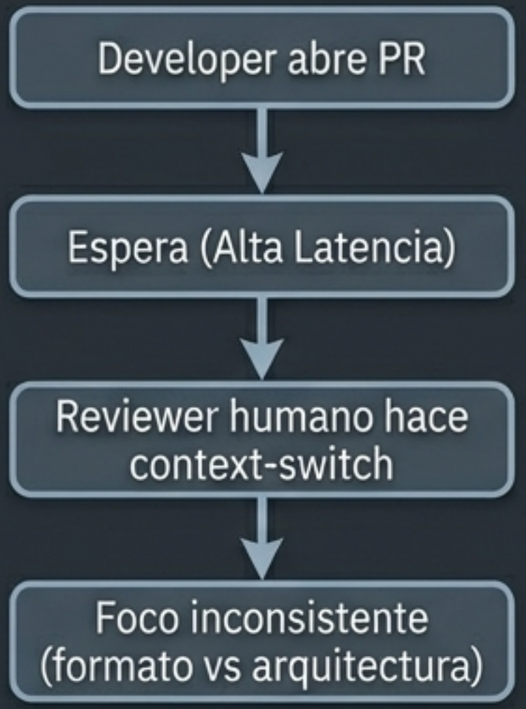


Ingesta de Contexto: Conexión GitHub MCP. Mediante la configuración en `config.yaml`, Hermes obtiene acceso de lectura nativo a **PRs, diffs** y tickets del repositorio sin requerir webhooks complejos.

Orquestación Temporal: Cron Scheduling Integrado. Programación en lenguaje natural (ej. 'Revisa la rama main por nuevos PRs cada 6 horas y ejecuta un code review'). El sistema crea y gestiona el cron job automáticamente.

Code Review Autónomo: Eliminando Latencia e Inconsistencia

Flujo Tradicional (Secuencial)

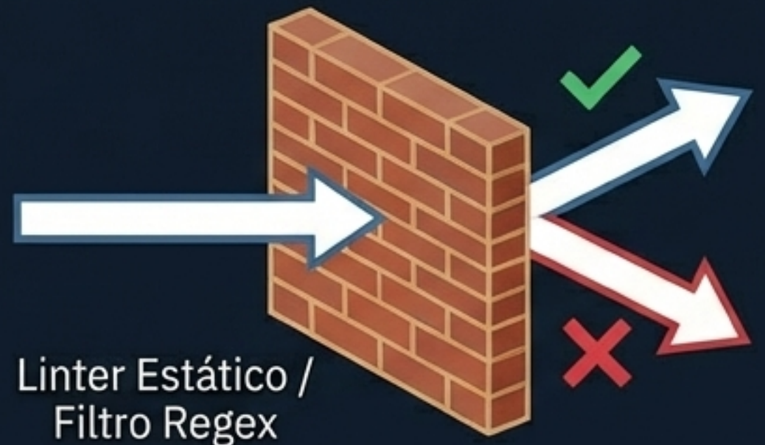


Flujo Hermes (Event-Driven Paralelo)

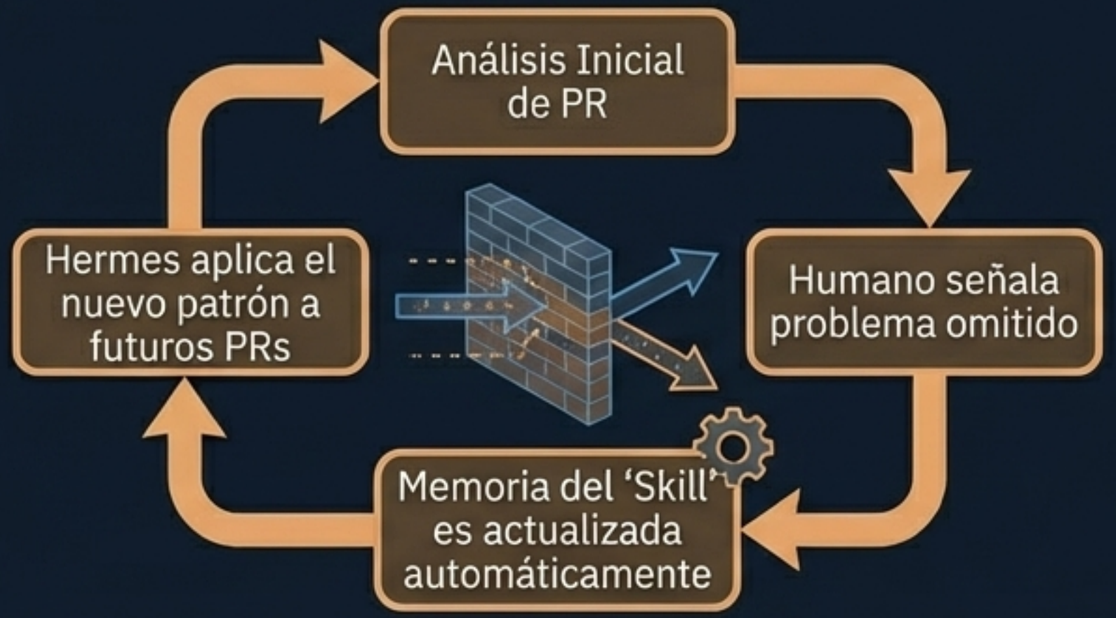
Mecanismo Base: En lugar de depender de la disponibilidad humana, Hermes intercepta el evento del PR y aplica reglas de manera determinista en paralelo.

Inyección de Reglas (Skills): Los estándares se configuran en lenguaje natural (ej. 'Las funciones deben tener <50 líneas', 'Manejo de errores con tipos personalizados').

Estándares Vivos vs. Análisis Estático Tradicional



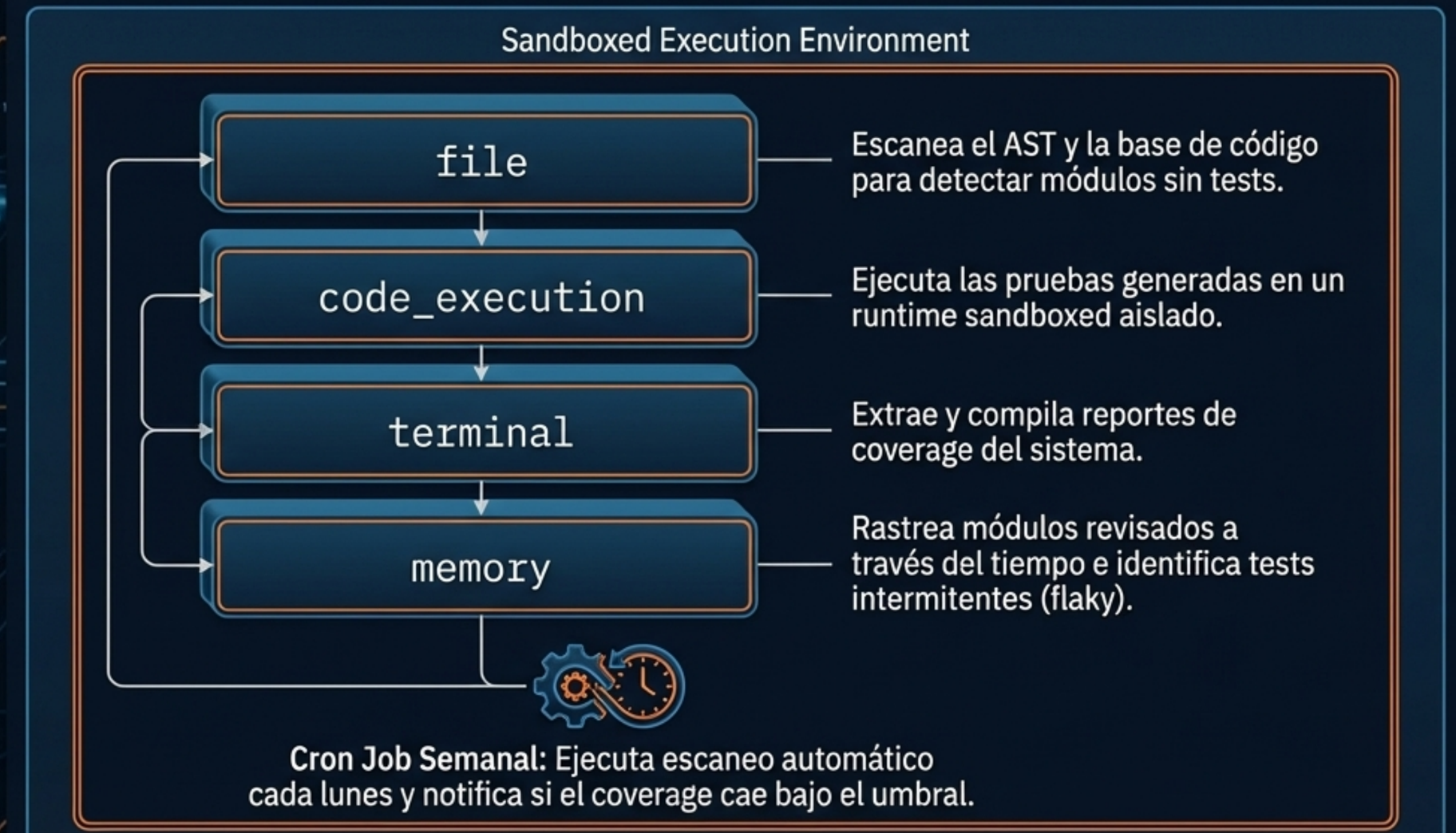
El Problema del Linter: Las reglas tradicionales son estáticas y rígidas, incapaces de entender el contexto arquitectónico.



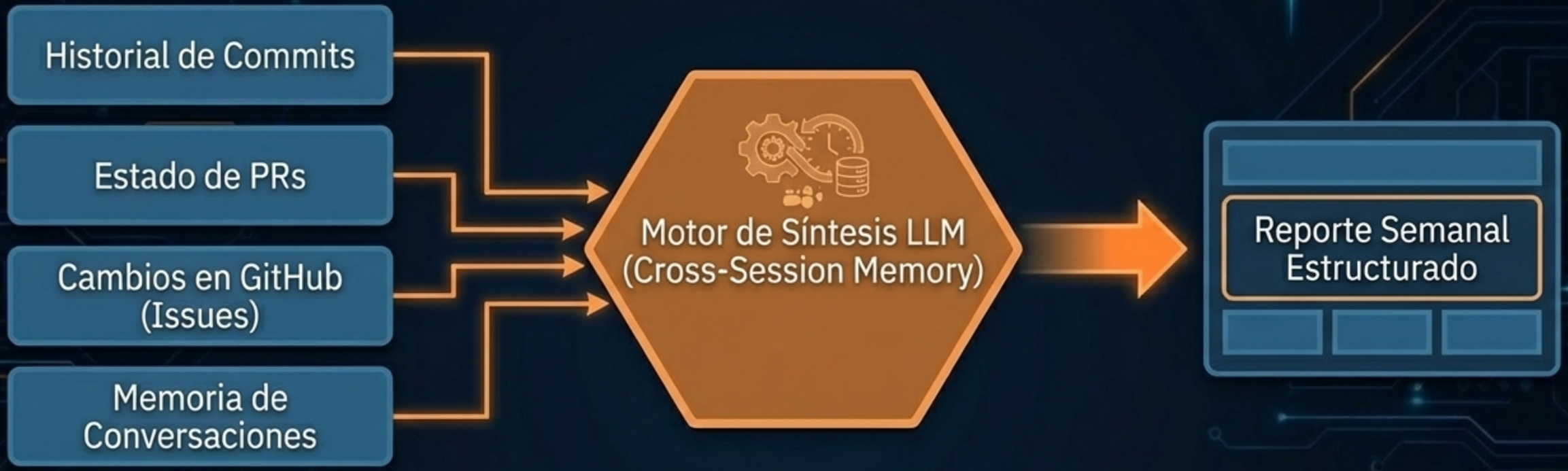
La Solución de Hermes: Los estándares de revisión son "vivos". Evolucionan automáticamente con el feedback del desarrollador interiorizando nuevos patrones arquitectónicos.

Arquitectura de Generación y Ejecución de Pruebas

Diferenciador: Mientras Claude Code es reactivo ("escribe pruebas para esta función"), Hermes es proactivo: descubre, escribe, ejecuta, corrige y reporta automáticamente.



Memoria de Estado y Síntesis de Reportes

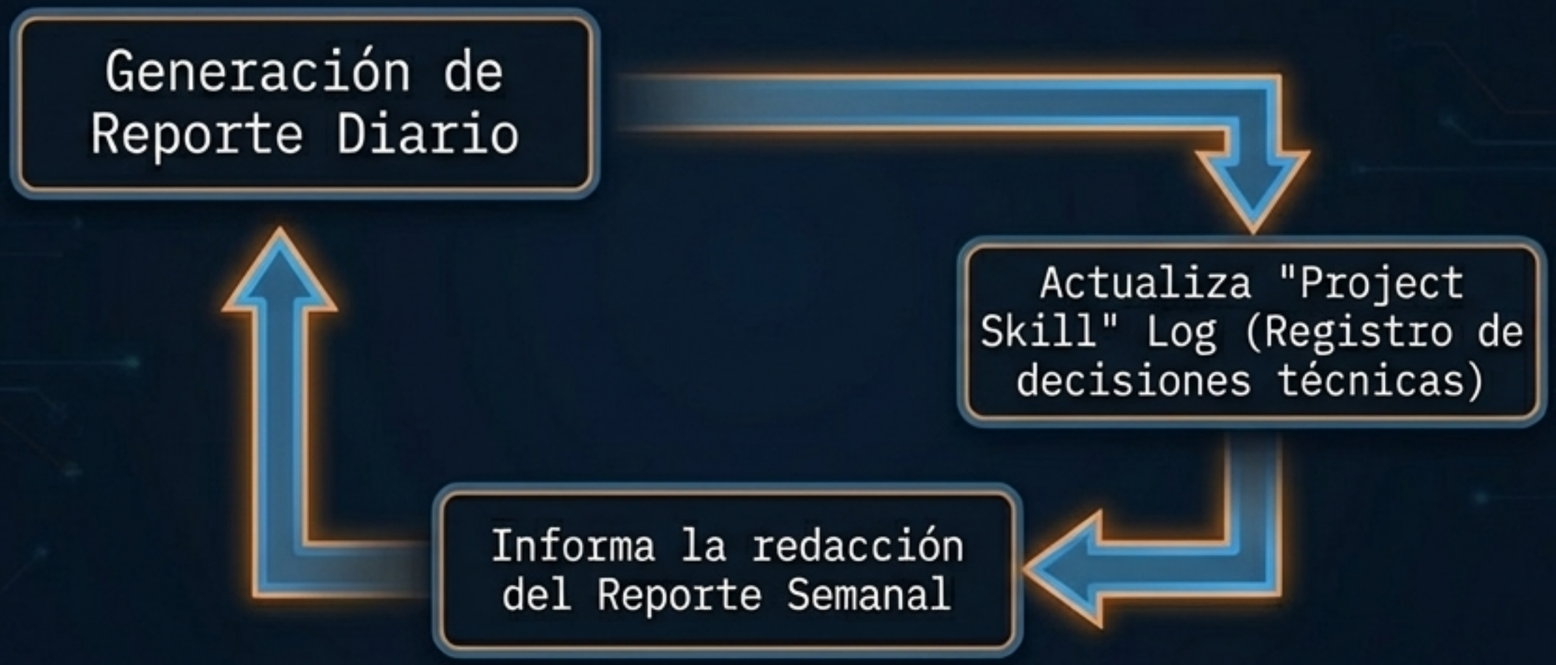


El Mecanismo
Hermes extrae datos crudos a través del GitHub MCP y los cruza con la 'Memoria de Conversación' de sesiones previas con el desarrollador.

El Salto Cualitativo
Los reportes no son listas de commits. Construyen una narrativa de causa y efecto temporal: comprende qué bug se resolvió el lunes, por qué cambió la arquitectura el miércoles, y la razón del rechazo de un PR el viernes.

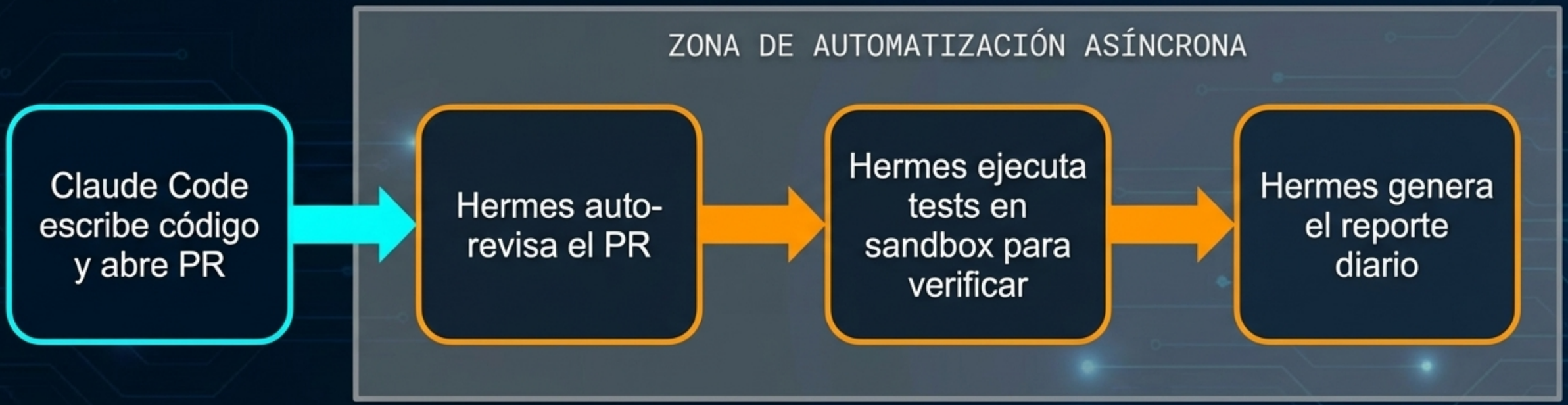
El Ciclo Virtuoso del Conocimiento Técnico (Best Practice)

Estrategia de Configuración: Configurar Hermes para que, al generar el reporte diario, actualice simultáneamente un Skill del Proyecto designado.



Resultado Arquitectónico: Al redactar el reporte semanal, Hermes consulta estas decisiones explícitamente en su memoria de estado, en lugar de intentar inferir la intención arquitectónica únicamente a partir de mensajes de commit crudos.

Síntesis: El Pipeline Multi-Agente CI/CD End-to-End



- El Cambio de Paradigma: Al implementar esta arquitectura, el enfoque del ingeniero transiciona de la ejecución (‘escribir código + revisar + probar + reportar’) a la verificación pura (‘escribir código + confirmar resultados’).
- Todo proceso intermediario entre la creación del PR y el reporte final de verificación es gestionado de manera autónoma y continua.

⚠ Límites Arquitectónicos y Fallbacks (Human-in-the-Loop)

1.	Limitación Declarada	La revisión de código de Hermes es un suplemento, no un reemplazo del juicio humano experto.
2.	Alcance Óptimo	Extremadamente eficaz para la detección de problemas basados en Extremadamente eficaz para la detección de problemas basados en patrones (nomenclatura inconsistente, falta de tests, casos límite no contemplados).
3.	Directriz de Seguridad	Las decisiones arquitectónicas fundamentales aún requieren revisión manual. Es obligatorio mantener un Human-in-the-Loop (HITL) para la validación de módulos core y lógica de negocio crítica.