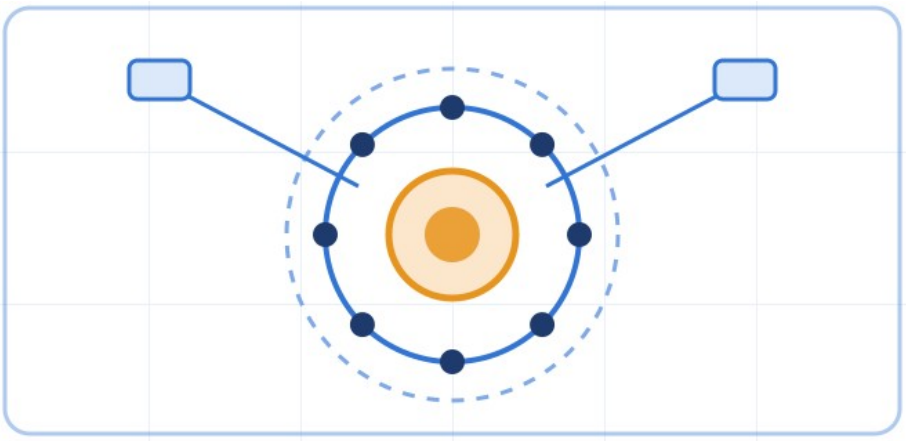
Scrum Manager®
Uncovering better ways

Guía

Harness Engineering

Desarrollo de los temas del State of the art



Actualizado a junio de 2026

Scrum Manager® · Skill Arena

Sobre este documento

Este documento desarrolla en profundidad los temas del mapa de referencia (State of the art) para adoptar agentes de IA con rigor profesional: recubrir un modelo con guías y sensores que conviertan su producción en trabajo fiable, verificable y útil, sea cual sea la disciplina, que está disponible en: Skill Arena.

Úsalo como material de consulta para mantener y contrastar tu práctica con el conocimiento profesional actual.

Se complementa con la plataforma de entrenamiento y evaluación en Skill Arena. En el área Harness Engineering puedes realizar pruebas de entrenamiento para contrastar y mejorar tu nivel de conocimiento y si lo deseas también puedes obtener un diploma que acredita curricularmente la solvencia y vanguardia profesional en esta área.

Estado del conocimiento

El conocimiento sobre Harness Engineering evoluciona de forma extremadamente rápida: la disciplina apenas existía hace año y medio y su vocabulario se ha estabilizado a lo largo de 2026 en torno a unas pocas publicaciones de referencia. Su núcleo conceptual (el marco de guías y sensores, los principios de contexto y verificación) ya está asentado, pero los patrones de la frontera —verificación sustantiva, arquitecturas multiagente, plantillas reutilizables— se redefinen casi cada mes. En los distintos apartados del documento, las etiquetas (ESTABLECIDO, EN CONSOLIDACIÓN, EMERGENTE) ayudan a identificar la madurez de cada concepto:

ESTABLECIDO consenso asentado; conocimiento que se da por necesario.

EN CONSOLIDACIÓN gana adopción con rapidez; aún no universal pero ya relevante.

EMERGENTE frontera reciente; alta relevancia y alta volatilidad.

Índice

Capítulos del desarrollo, en el orden del State of the art.

BLOQUE A · EL CAMBIO DE PARADIGMA

Capítulo 1. Inteligencia frente a agencia

ESTABLECIDO

La inteligencia la entrena el fabricante; la agencia la construyes tú alrededor. Distinguirlas es el punto de partida de toda la disciplina.

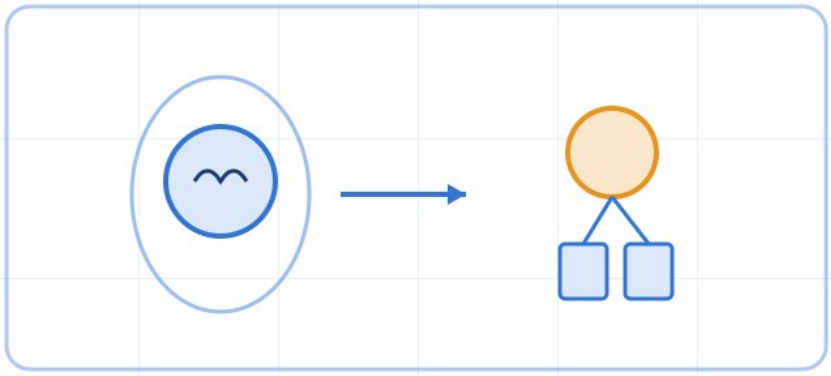
Introducción

Hay una distinción que parece sutil y resulta ser el cimiento de todo lo demás: la diferencia entre inteligencia y agencia. La inteligencia es una propiedad entrenada del modelo: su capacidad de razonar sobre el lenguaje, conectar conceptos, sintetizar información, planificar e intuir patrones. La agencia es algo distinto: la capacidad de actuar sobre el mundo —leer documentos, ejecutar acciones, modificar artefactos, invocar herramientas, mantener coherencia entre sesiones y producir entregables que alguien pueda verificar.

Un modelo dotado de mucha inteligencia pero sin entorno operativo es como un cerebro flotando en un tarro: brillante, capaz de razonar y opinar, pero incapaz de comprobar nada o de dejar una huella verificable en un proyecto real. Para que esa inteligencia se convierta en un agente útil para trabajo profesional necesita un cuerpo. Ese cuerpo es el arnés.

1.1 Por qué el modelo, solo, no basta

Cuando se encarga una tarea sustantiva a un modelo de frontera sin nada alrededor, el patrón de fallo es reconocible en cualquier disciplina. El resultado llega pulido, articulado y persuasivo, y a la vez plagado de problemas que no se ven a primera vista: fuentes inventadas, cifras aproximadas presentadas como exactas, decisiones que no sobreviven a una revisión seria, o código que parece funcionar y rompe cosas invisibles. El modelo empieza rápido, dice cosas inteligentes y, sin forma de corregir el curso a tiempo, termina entregando algo plausible pero hueco.



La inteligencia sola razona pero no actúa; con arnés, el modelo opera sobre artefactos y deja trabajo verificable.

La oscilación entre lo deslumbrante y lo desesperante que muchos profesionales experimentan al usar agentes no es, por tanto, un defecto del modelo: es el síntoma de que falta arnés alrededor.

1.2 Qué cambia cuando hay agencia bien gobernada

El mismo modelo, recubierto de guías que orientan su acción antes de actuar y de sensores que detectan errores después, completa tareas reales con calidad de producción. Es más lento turno a turno, porque verifica y se corrige, pero entrega valor real y aguanta la revisión de un experto humano. La agencia no es una propiedad mágica del modelo más nuevo: es el resultado de construir el entorno correcto a su alrededor.

La idea en una frase. La inteligencia se compra; la agencia útil se construye. Dos profesionales con el mismo modelo entregan trabajo de calidad muy distinta, y la diferencia está siempre en lo que rodea al modelo, no en el modelo.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Distinguir inteligencia (propiedad entrenada del modelo) de agencia (capacidad de actuar sobre el mundo) y explicar por qué son independientes.
- Identificar en un resultado concreto los síntomas de ausencia de arnés: fuentes sin verificar, cifras sin fuente, conclusiones que no resisten revisión.
- Atribuir la oscilación entre resultados deslumbrantes y desesperantes a la falta de arnés, en lugar de a una limitación del modelo.
- Justificar por qué un modelo más capaz no resuelve, por sí solo, un problema de agencia mal gobernada.

Errores y antipatrones frecuentes

Culpar al modelo. Atribuir cada fallo a que «el modelo no es lo bastante bueno» y esperar a la siguiente versión, en lugar de examinar qué falta en el entorno.

Confundir fluidez con corrección. Aceptar un resultado porque está bien escrito y articulado, sin comprobar si las afirmaciones se sostienen.

Pedir tareas sustantivas sin cuerpo. Encargar trabajo crítico a un modelo sin instrucciones, materiales ni verificaciones, y sorprenderse del resultado plausible pero hueco.

Para profundizar

· Böckeler · Harness engineering (martinfowler.com)

· Mollick · One Useful Thing

BLOQUE A · EL CAMBIO DE PARADIGMA

Capítulo 2. La ecuación Agente = Modelo + Arnés

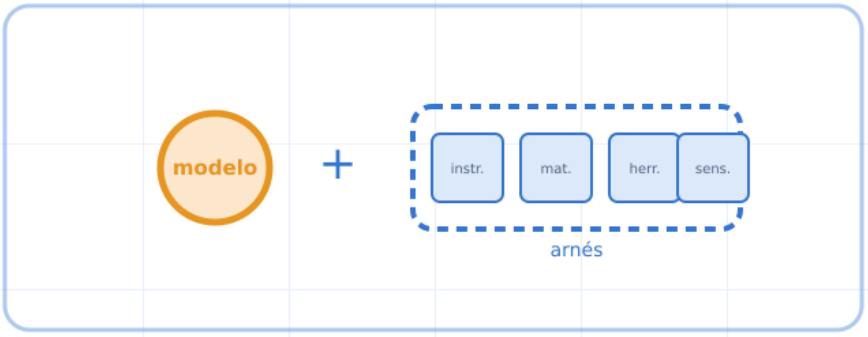
ESTABLECIDO

La fórmula canónica de la disciplina: el arnés es todo lo que no es el modelo. El modelo es un commodity; el arnés es la ventaja competitiva.

Introducción

En abril de 2026, Birgitta Böckeler, Distinguished Engineer en Thoughtworks, publicó en el blog de Martin Fowler el artículo que dio a este campo su vocabulario común. Aunque su título hablaba de agentes de codificación, el marco conceptual aplica a cualquier trabajo asistido por agentes y se convirtió rápidamente en la lingua franca de la disciplina. Su aportación más citada es una ecuación deliberadamente simple.

Agente = Modelo + Arnés.



El arnés engloba todo lo que no es el modelo: instrucciones, materiales, herramientas, permisos, orquestación, memoria y sensores.

2.1 Qué engloba el arnés

El arnés es todo lo que no es el modelo. En la práctica, eso incluye las instrucciones que el agente lee al empezar, los documentos a los que tiene acceso, las herramientas que puede invocar, las políticas de permisos que defines para él, los bucles de orquestación que lo activan, la memoria que persiste entre sesiones, los sensores que verifican su trabajo y las interfaces de control humano. Es una definición ancha a propósito: cualquier pieza que recubra al modelo y condicione su comportamiento forma parte del arnés.

2.2 El modelo es un commodity; el arnés es la ventaja

De la ecuación se sigue una afirmación sencilla y agresiva: si el modelo es accesible para cualquiera, la diferencia entre el profesional que entrega trabajo riguroso y el que entrega humo articulado no puede estar en el modelo. Está en el arnés. Esto reordena dónde invertir el esfuerzo: no en perseguir el modelo más nuevo, sino en construir las instrucciones, los materiales, las verificaciones y las plantillas que convierten ese modelo en un agente fiable.

Por qué importa el vocabulario. Sin un modelo mental compartido de qué es y qué no es el arnés, las prácticas concretas que vienen después no tienen dónde anclarse. La ecuación es la pieza que permite hablar de todo lo demás con precisión.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Enunciar la ecuación Agente = Modelo + Arnés y explicar qué incluye el término «arnés».
- Clasificar elementos concretos de su entorno de trabajo como parte del modelo o parte del arnés.
- Argumentar por qué la ventaja competitiva reside en el arnés y no en el modelo, dado que el modelo es accesible para todos.
- Reorientar la inversión de esfuerzo desde «conseguir el mejor modelo» hacia «construir el mejor arnés».

Errores y antipatrones frecuentes

Reducir el arnés al prompt. Creer que el arnés es solo «escribir buenas instrucciones», ignorando materiales, herramientas, permisos, memoria y sensores.

Esperar que el modelo nuevo lo resuelva. Posponer la construcción del arnés a la espera de un modelo más capaz que haga innecesario el esfuerzo.

No nombrar las piezas. Trabajar con un arnés implícito que nadie ha descrito, lo que impide razonar sobre qué falta o qué falla.

Para profundizar

· Böckeler · Harness engineering (martinfowler.com)

· Thoughtworks · What is harness engineering

BLOQUE A · EL CAMBIO DE PARADIGMA

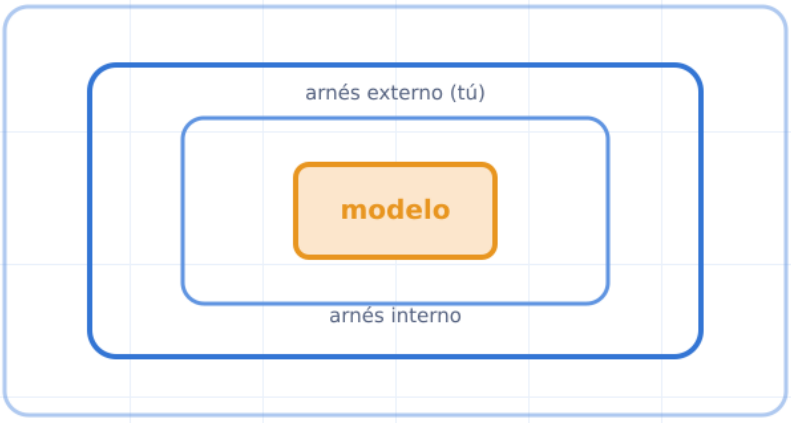
Capítulo 3. Arnés interno frente a arnés externo

ESTABLECIDO

El fabricante construye el arnés interno; tú construyes el externo sobre él. El grueso del valor que extraerás está en el externo.

Introducción

La palabra «arnés» nombra al menos dos cosas distintas según el contexto, y confundirlas dispersa el esfuerzo. Böckeler las dibuja como círculos concéntricos: en el centro, el modelo; alrededor, el arnés interno que construye el fabricante; y fuera, el arnés externo que construyes tú.



Tres círculos concéntricos: el modelo, el arnés interno del fabricante y el arnés externo que construyes tú.

3.1 El arnés interno

El arnés interno lo construyen los fabricantes alrededor del modelo. Incluye la interfaz de chat, el sistema de proyectos, los conectores nativos, la memoria integrada, la búsqueda, los modos especializados, el bucle de orquestación, los ejecutores de herramientas y los sistemas de permisos. Tú usas este arnés —y conviene conocer sus capacidades para aprovecharlo—, pero no lo construyes ni lo modificas.

3.2 El arnés externo

El arnés externo lo construyes tú sobre el interno que te dan hecho. Es lo que escribes en los archivos de instrucciones del proyecto, los documentos que pones a su disposición, las habilidades especializadas que defines, los conectores que enchufas, las verificaciones y rutinas que aplicas y las plantillas que diseñas. Esta guía cubre exclusivamente el arnés externo.

La buena noticia. El grueso del valor que un profesional extrae de los agentes en su trabajo diario proviene de hacer bien el arnés externo, no de comprender el interno. Saber dónde acaba uno y empieza el otro evita gastar energía donde no se puede intervenir.

3.3 Por qué la distinción es práctica, no académica

Delimitar responsabilidades cambia las decisiones del día a día. Cuando algo falla, la primera pregunta es de qué círculo es el problema: si es del arnés interno, la vía es elegir mejor la herramienta o ajustar su configuración; si es del externo, la vía es modificar lo que tú controlas. Mezclar ambos lleva a intentar arreglar con prompts cosas que pertenecen a la herramienta, o a cambiar de herramienta cuando el problema estaba en unas instrucciones mal escritas.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Distinguir el arnés interno (responsabilidad del fabricante) del externo (responsabilidad propia) usando el modelo de círculos concéntricos.
- Clasificar una capacidad concreta de su entorno como parte del arnés interno o del externo.
- Concentrar el esfuerzo de mejora en el arnés externo, donde reside la mayor parte del valor recuperable.
- Ante un fallo, determinar si la palanca de corrección está en la herramienta elegida o en el arnés que uno mismo construye.

Errores y antipatrones frecuentes

Querer reconstruir el arnés interno. Invertir tiempo en replicar la gestión de contexto o la orquestación del fabricante, en lugar de aprovechar lo que ya viene hecho.

Atribuir a la herramienta fallos del arnés externo. Cambiar de producto buscando una solución a un problema que estaba en las propias instrucciones o materiales.

Ignorar las capacidades del arnés interno. Construir externamente algo que la herramienta ya ofrece de fábrica, duplicando esfuerzo.

Para profundizar

· Böckeler · Harness engineering (martinfowler.com)

· Anthropic · Effective harnesses for long-running agents

BLOQUE B · LAS DOS COLUMNAS: GUÍAS Y SENSORES

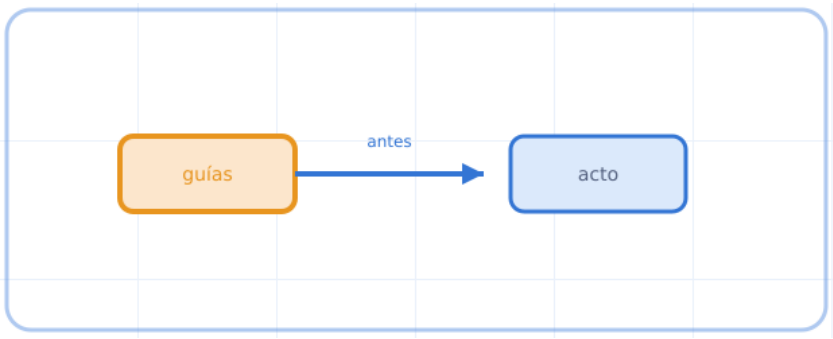
Capítulo 4. Guías: controles antes del acto

ESTABLECIDO

Las guías anticipan el comportamiento del agente y lo orientan antes de actuar, aumentando la probabilidad de acertar a la primera.

Introducción

El marco de Böckeler descompone el arnés externo en dos tipos de control que actúan en momentos distintos del trabajo del agente. Las guías son la mitad que actúa antes. En la terminología cibernética que toma prestada el campo, son controles feedforward: anticipan lo que va a pasar e intentan dirigirlo, en lugar de reaccionar una vez ha pasado.



Una guía actúa antes del acto: orienta al agente para que acierte a la primera.

4.1 Qué es una guía

Una guía es todo lo que el agente lee al empezar su trabajo y lleva consigo mientras opera. Su función es aumentar la probabilidad de que acierte a la primera, reduciendo la necesidad de corregir después. No garantiza el acierto —para eso están los sensores—, pero parte al agente de un estado conocido y alineado con tus reglas.

4.2 Tipos típicos de guía

Las guías adoptan formas reconocibles en cualquier disciplina:

- Archivos de instrucciones que describen el proyecto, sus convenciones y sus reglas.
- Documentación de arquitectura, de método o de proceso.
- Glosarios terminológicos que fijan el vocabulario.
- Guías de estilo: de código, editoriales, de citación.
- Plantillas estructurales para los entregables (la forma canónica de un módulo, un informe, un memorando, una pieza).
- Corpus curados de referencia: aplicaciones modelo, bibliografías acotadas, ejemplos canónicos.
- Habilidades («skills») reutilizables que el agente invoca cuando el contexto lo pide.

El estándar emergente. Buena parte del ecosistema ha convergido en un formato abierto para el archivo de instrucciones, AGENTS.md, que se desarrolla en el capítulo 13. Lo relevante aquí es que una guía bien hecha es legible, operativa y del tamaño justo.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Definir una guía como control feedforward y explicar en qué momento del ciclo del agente actúa.
- Reconocer y enumerar los tipos típicos de guía en su propia disciplina.
- Diferenciar la función de una guía (orientar antes) de la de un sensor (verificar después).
- Seleccionar qué guías necesita un proyecto concreto según sus convenciones y entregables.

Errores y antipatrones frecuentes

Confundir guía con verificación. Esperar que un archivo de instrucciones detecte errores; las guías orientan, no comprueban.

Guías genéricas. Escribir reglas tan amplias («sé riguroso», «escribe bien») que no orientan ninguna decisión concreta.

Guías que nadie mantiene. Dejar que la documentación de método envejezca hasta describir un proceso que ya no se sigue.

Para profundizar

· Böckeler · Harness engineering (martinfowler.com)

· AGENTS.md · estándar abierto

BLOQUE B · LAS DOS COLUMNAS: GUÍAS Y SENSORES

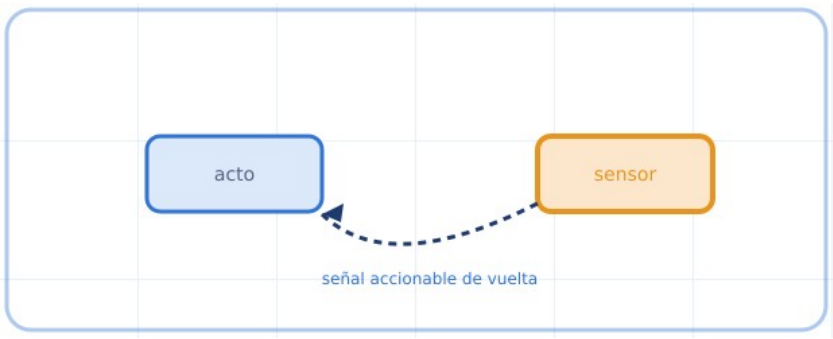
Capítulo 5. Sensores: controles después del acto

ESTABLECIDO

Los sensores observan al agente después de producir algo y le permiten autocorregirse, devolviendo señales accionables que otra IA puede consumir.

Introducción

Los sensores son la otra mitad del marco: actúan después del acto. Son controles feedback: observan lo que el agente ha producido y le devuelven una señal que le permite corregirse. Lo más potente de un buen sensor es que su señal no es un veredicto opaco, sino una instrucción de reparación.



Un sensor actúa después del acto y devuelve una señal accionable que el agente puede usar para autocorregirse.

5.1 La señal accionable

Un sensor mediocre falla con un mensaje críptico. Un buen sensor devuelve algo como: «la afirmación X falla la verificación Y; el valor esperado era A y el observado es B; la causa probable está en Z; corrige así o así». La diferencia no es cosmética: la segunda versión enseña al agente a arreglarlo, y por eso multiplica la eficacia del bucle de autocorrección. Un mensaje de error es, en realidad, un prompt que estás inyectando al agente.

5.2 Tipos típicos de sensor

Como las guías, los sensores adoptan formas reconocibles:

- Suites de verificaciones automáticas: linters, tests, type checkers, comprobadores de coherencia.
- Listas de comprobación pre-entrega: ¿está todo lo requerido?, ¿están las afirmaciones soportadas?, ¿hay coherencia interna?
- Fact-checking asistido por un segundo agente y verificación de citas contra fuentes reales.
- Análisis estructural o estilístico: complejidad, legibilidad, detección de duplicados.
- Revisores adversariales: un segundo agente cuyo único trabajo es objetar.
- Verificaciones end-to-end: un test que ejerce el sistema completo; una revisión humana en la última milla.

Por qué hacen falta los dos. Solo con guías, el agente codifica reglas pero nunca sabe si su producción las cumple, y repite errores estructuralmente válidos pero rotos. Solo con sensores, recibe feedback sin saber a dónde iba y da bandazos. Con ambos, sabe a dónde va y se corrige cuando se desvía.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Definir un sensor como control feedback y situar el momento en que actúa.
- Redactar un mensaje de sensor accionable que indique qué falló, qué se esperaba y cómo corregirlo.
- Enumerar tipos de sensor aplicables a su disciplina, tanto automáticos como humanos.
- Justificar por qué un arnés necesita guías y sensores a la vez, y qué falla si tiene solo uno de los dos.

Errores y antipatrones frecuentes

Mensajes de error opacos. Sensores que dicen «hay problemas, revisa» sin localizar el fallo ni sugerir corrección, desperdiciando el bucle de feedback.

Verificar solo al final. Pasar los sensores una sola vez, al cierre, en lugar de en el momento en que el agente comete la violación.

Arnés solo con sensores. Renunciar a las guías y dejar que el agente descubra las reglas a base de fallar, lo que es lento y errático.

Para profundizar

· Böckeler · Maintainability sensors for coding agents

· Böckeler · Harness engineering (martinfowler.com)

BLOQUE B · LAS DOS COLUMNAS: GUÍAS Y SENSORES

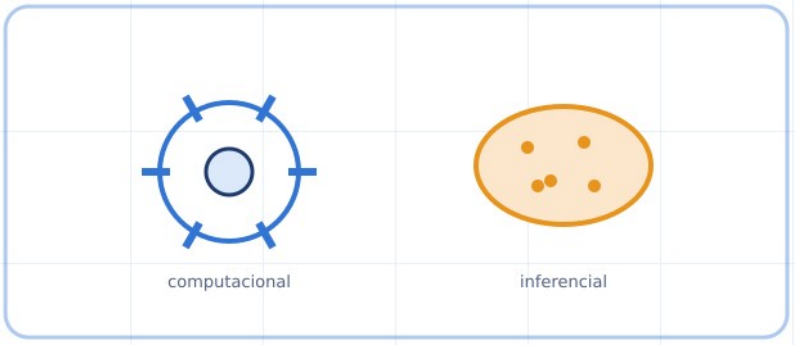
Capítulo 6. Computacional frente a inferencial

ESTABLECIDO

Todo control se ejecuta de dos maneras. La regla pragmática: siempre que puedas resolver algo con un control computacional, no uses uno inferencial.

Introducción

Cualquier control —sea guía o sensor— se ejecuta de una de dos maneras según quién lo aplique. Esta clasificación, ortogonal a la de guías y sensores, decide buena parte del coste y la fiabilidad de un arnés.



El control computacional es determinista, rápido y barato; el inferencial es potente para semántica pero lento, caro y probabilístico.

6.1 Control computacional

Lo ejecuta una herramienta determinista: un linter, un test, un script, una búsqueda en una base de datos. Es rápido (de milisegundos a segundos), de coste mínimo y totalmente fiable dentro de su alcance. No miente: si un test pasa, pasa. Su límite es que solo verifica lo que puede expresarse de forma mecánica.

6.2 Control inferencial

Lo ejecuta otro modelo de lenguaje, en el papel de juez o evaluador (el patrón conocido como LLM-as-judge). Es potente para juicios semánticos que ninguna herramienta determinista alcanza —¿este resultado está sobreingeniado?, ¿este texto matiza correctamente?, ¿esta conclusión es defendible?—, pero es lento, caro y probabilístico: puede equivocarse, y no de forma reproducible.

6.3 La regla pragmática

Siempre que algo pueda resolverse con un control computacional, no uses uno inferencial. Un arnés maduro tiene los dos tipos, en el orden correcto: primero los computacionales, que filtran barato y fiable; después los inferenciales, para lo que aquellos no alcanzan. Un arnés inmaduro descansa todo en controles inferenciales porque parecen más sofisticados, y termina costando una fortuna sin entregar fiabilidad.

Ejemplo transversal. Verificar que cada cifra de un informe tiene una fuente adyacente es computacional: un script que escanea dígitos y marca los que no la tienen. Juzgar si la interpretación de esa cifra es razonable es inferencial. El primero es barato y seguro; reserva el segundo para lo que de verdad lo necesita.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Distinguir un control computacional de uno inferencial según quién lo ejecuta y con qué propiedades de velocidad, coste y fiabilidad.
- Decidir, ante una verificación concreta, si debe resolverse de forma computacional o inferencial.
- Ordenar los controles de un arnés colocando los computacionales como primera línea y los inferenciales como segunda.
- Reconocer un arnés inmaduro por su dependencia excesiva de controles inferenciales.

Errores y antipatrones frecuentes

Inferencial por defecto. Usar otro modelo como juez para tareas que un script resolvería más rápido, más barato y sin error.

Confiar el cierre a un solo LLM. Descansar toda la verificación en un revisor inferencial, con su coste alto y sus falsos positivos y negativos.

Ignorar el límite del computacional. Pretender que un linter capture juicios semánticos que requieren interpretación.

Para profundizar

· Böckeler · Harness engineering (martinfowler.com)

· Böckeler · Maintainability sensors for coding agents

BLOQUE B · LAS DOS COLUMNAS: GUÍAS Y SENSORES

Capítulo 7. Las tres dimensiones de regulación

EN CONSOLIDACIÓN

El arnés regula la producción del agente en tres dimensiones: calidad estructural, forma profesional del oficio y comportamiento funcional.

Introducción

Böckeler propone ver el arnés como un gobernador que regula la producción del agente hacia un estado deseado. Pero «el estado deseado» no es una sola cosa: son al menos tres dimensiones, con grados de madurez muy distintos. Saber en cuál opera cada verificación evita el desequilibrio típico de los arneses reales.

Tres dimensiones de regulación: la estructural es la más madura; la funcional sigue siendo la frontera abierta.

7.1 Calidad estructural

Regula la calidad interna del producto: legibilidad, estructura, coherencia, consistencia, ausencia de redundancias, integridad de las partes. Es la dimensión más madura, porque hereda décadas de tooling y manuales de estilo. Aquí caben sensores computacionales (consistencia, patrones prohibidos, complejidad, duplicados, comprobar que todas las secciones requeridas existen) e inferenciales (revisión de tono, detección de soluciones semánticamente redundantes). Böckeler ha empezado a operacionalizar esta dimensión con catálogos concretos de sensores de mantenibilidad.

7.2 Forma profesional del oficio

Regula los estándares transversales del trabajo: rigor metodológico, neutralidad, gestión de la incertidumbre, transparencia sobre fuentes y supuestos, observabilidad, cumplimiento normativo cuando aplica. Es la idea de las fitness functions aplicada al trabajo agéntico. Ejemplos: una habilidad que verifica que cada afirmación cuantitativa va con su fuente, periodo y método; una plantilla que obliga a incluir una sección de limitaciones en todo entregable; una verificación de que la evidencia contradictoria se expone en lugar de esconderse.

7.3 Comportamiento funcional

Es el elefante en la habitación: ¿el entregable resuelve realmente la pregunta?, ¿el código hace lo que necesitas?, ¿la conclusión es defendible y las recomendaciones ejecutables? Hoy la mayoría confía en una especificación previa y en verificaciones que a menudo genera el propio agente, complementadas con revisión humana. Eso pone demasiada fe en comprobaciones producidas por la misma IA que hizo el trabajo. Es la frontera abierta del campo y el objeto del capítulo 16.

La consecuencia práctica. Cuanto más crítica sea la corrección sustantiva de un entregable, menos autonomía debe darse al agente y más rol humano debe haber. El desequilibrio típico: casi todo el tooling cubre la dimensión estructural, mientras la funcional queda infraservida.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Nombrar las tres dimensiones de regulación y describir qué controla cada una.
- Clasificar una verificación concreta en la dimensión que regula.
- Identificar cuál de las tres dimensiones está infraservida en un arnés dado, normalmente la funcional.
- Ajustar el grado de autonomía del agente en función de cuán crítica sea la corrección sustantiva del entregable.

Errores y antipatrones frecuentes

Cubrir solo lo estructural. Llenar el arnés de verificaciones de forma y coherencia y descuidar si el entregable hace lo que debía.

Confundir bien formado con correcto. Aceptar un entregable porque pasa los sensores estructurales, sin comprobar la dimensión funcional.

Misma autonomía para todo. Dar al agente el mismo margen en una tarea trivial que en una de alta consecuencia sustantiva.

Para profundizar

· Böckeler · Maintainability sensors for coding agents

· Böckeler · Harness engineering (martinfowler.com)

BLOQUE C · PRINCIPIOS QUE GOBIERNAN EL ARNÉS

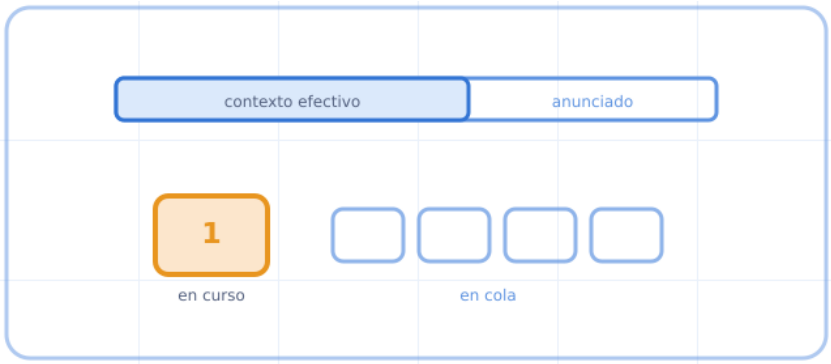
Capítulo 8. WIP = 1 y el presupuesto de contexto

ESTABLECIDO

Cada sesión trabaja en una sola tarea hasta verificarla. El motivo es físico: el contexto utilizable del modelo es mucho menor que el anunciado.

Introducción

Work-in-Progress igual a uno es la regla operativa con mayor impacto empírico en la fiabilidad del trabajo agéntico: cada sesión trabaja en una sola tarea o entregable y no avanza al siguiente hasta que el actual está verificado. No es una preferencia estilística; tiene una raíz medible en cómo funcionan los modelos.



El contexto efectivo es menor que el anunciado; mantener una sola tarea en curso protege ese presupuesto limitado.

8.1 Por qué el contexto es un presupuesto, no un almacén

La ventana de contexto de un modelo es finita y, peor, su parte utilizable es bastante menor que la anunciada. El benchmark RULER de NVIDIA, diseñado para medir el contexto efectivo real, muestra que los modelos dejan de mantener su rendimiento mucho antes de llenar la ventana nominal. A esto se suma el efecto Lost in the Middle (Liu et al., 2023): los modelos pierden eficacia procesando información situada en el centro de contextos largos. El contexto se comporta como un presupuesto que se agota, no como un almacén que se llena sin coste.

8.2 Qué protege WIP = 1

Limitar el trabajo a una tarea por sesión protege ese presupuesto y reduce la carga cognitiva del agente. Las implementaciones reales muestran tasas de éxito notablemente más altas bajo WIP = 1 que en sesiones sin restricciones, donde el agente intenta varias tareas a la vez, satura el contexto y empieza a dar bandazos. En la práctica: no pidas tres tareas relacionadas en una sola sesión; cierra y abre una nueva.

El coste de refundamentar es minúsculo. Sí, abrir una sesión nueva obliga a «refundamentar» al agente leyendo los archivos del proyecto. Pero ese coste es ínfimo comparado con el de un agente confundido en mitad de un contexto saturado. El capítulo 14 explica cómo hacer esa refundamentación barata.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Enunciar la regla WIP = 1 y explicar su fundamento en el contexto efectivo y el efecto Lost in the Middle.
- Estimar que el contexto utilizable es menor que el anunciado y planificar el trabajo en consecuencia.
- Descomponer un encargo grande en tareas que quepan, de una en una, en una sesión.
- Decidir cuándo cerrar una sesión y abrir otra en lugar de encadenar tareas en la misma.

Errores y antipatrones frecuentes

Encadenar tareas en una sesión. Pedir varias tareas relacionadas de una vez para «ahorrar», saturando el contexto y degradando el resultado.

Tratar la ventana como infinita. Asumir que cabe todo porque el modelo anuncia una ventana enorme, ignorando el contexto efectivo real.

Temer el coste de reabrir. Evitar cerrar una sesión por no «perder» contexto, cuando refundamentar desde los archivos es barato.

Para profundizar

· Liu et al. · Lost in the Middle (arXiv)

· NVIDIA RULER (arXiv)

· Anthropic · Effective context engineering

BLOQUE C · PRINCIPIOS QUE GOBIERNAN EL ARNÉS

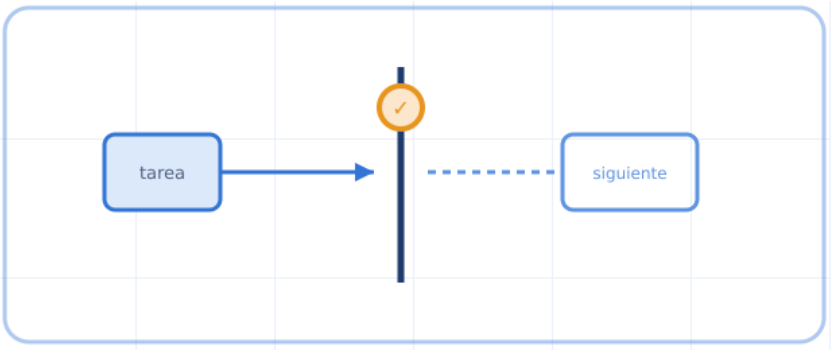
Capítulo 9. Pass-state gating y la brecha de verificación

ESTABLECIDO

«Hecho» no es un adjetivo, sino un resultado de verificaciones. El pase verificado prohíbe avanzar hasta que el entregable actual ha pasado sus comprobaciones.

Introducción

Existe una distancia peligrosa entre el sentimiento de confianza que produce un resultado bien presentado y su corrección real. Los agentes son optimistas estructurales: tienden a declarar «completo» lo que solo es «plausible». Esta brecha de verificación es la causa del fallo más caro del trabajo agéntico, y el arnés tiene un mecanismo directo para cerrarla.



El pase verificado funciona como una puerta: no se avanza al siguiente entregable sin verificación verde del actual.

9.1 La brecha de verificación

Preguntado «¿has terminado?», un agente responde «sí» más a menudo de lo que debería. No miente: su sesgo estructural lo empuja a cerrar. El resultado es el agente que entrega al 30% con plena confianza, o el informe con errores de atribución presentado como definitivo. La regla de oro que corrige este sesgo es simple: «hecho» no es un adjetivo; es un resultado de verificaciones.

9.2 El pase verificado

El pass-state gating materializa esa regla: el arnés prohíbe al agente avanzar al siguiente entregable hasta que el actual ha superado un conjunto explícito de verificaciones —tests verdes, fact-check pasado, citas verificadas, coherencia interna correcta, todas las secciones presentes, build correcto, smoke test correcto, lo que sea pertinente a cada tipo de trabajo. La puerta no abre sin la señal verde.

Default-FAIL. Un refinamiento útil del patrón: cada criterio empieza en «falso» y el agente no puede marcarlo como cumplido sin aportar evidencia primero. Invierte la carga de la prueba: en lugar de asumir que está bien salvo que se demuestre lo contrario, se asume que no lo está hasta que se demuestra que sí.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Explicar la brecha de verificación y el sesgo de los agentes a declarar completo lo plausible.
- Aplicar la regla «hecho es un resultado de verificaciones» en lugar de aceptar la autodeclaración del agente.

- Definir un conjunto explícito de verificaciones que un entregable debe pasar antes de considerarse cerrado.
- Implementar un pase verificado que impida avanzar sin evidencia, idealmente con criterios que parten de «no cumplido».

Errores y antipatrones frecuentes

Confianza prematura. Aceptar el «hecho» del agente sin verificar, y descubrir días después que media tarea estaba rota.

Verificaciones implícitas. No declarar qué debe pasar un entregable, dejando que «terminado» signifique cosas distintas en cada sesión.

Evidencia opcional. Permitir marcar criterios como cumplidos sin aportar prueba, lo que reabre la brecha de verificación.

Para profundizar

· Anthropic · Effective harnesses for long-running agents

· Anthropic · cwc-long-running-agents (GitHub)

BLOQUE C · PRINCIPIOS QUE GOBIERNAN EL ARNÉS

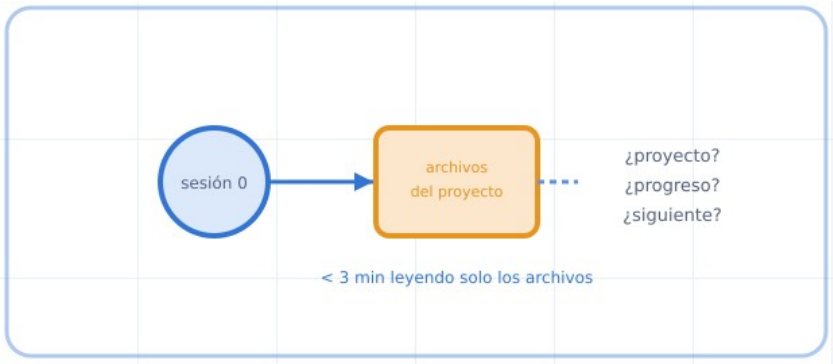
Capítulo 10. Cold-start test y legibilidad del entorno

EN CONSOLIDACIÓN

Abre una sesión nueva sin historial: ¿puede el agente, leyendo solo los archivos del proyecto, saber qué es, qué falta y cuál es el siguiente paso?

Introducción

La métrica más reveladora de la calidad de un arnés no mide al modelo, sino al entorno. El cold-start test consiste en abrir una sesión totalmente nueva, sin historial ni contexto previo, y comprobar si el agente, leyendo solo los archivos del proyecto, identifica qué es el sistema, cuál es el progreso actual y qué falta por hacer.



Una sesión sin historial debería orientarse leyendo solo los archivos del proyecto, en pocos minutos.

10.1 Cómo se interpreta

Si el agente se orienta en pocos minutos, el arnés es sólido. Si tarda mucho, probablemente documenta demasiado o lo equivocado. Si no lo logra, el conocimiento vive en la cabeza de un humano y no en el proyecto: el arnés está roto. Para proyectos que viven semanas o meses con sesiones intercaladas, este indicador es decisivo, porque sin él cada sesión empieza con un largo «ponerse al día».

10.2 El corolario: lo que no está, no existe

De aquí se sigue una regla incómoda: lo que no está en el proyecto no existe para el agente. El conocimiento tácito —lo hablado en una reunión, los porqués que solo conoce el experto, los gotchas que nadie ha escrito— es invisible. El trabajo del Harness Engineer es codificar ese conocimiento tácito en artefactos explícitos: briefs, registros de decisiones, glosarios, plantillas, comentarios, tests que documentan casos límite. Si se acordó «no tocamos X» o «siempre hacemos Y», ese acuerdo tiene que estar en un archivo, no solo en tu memoria.

10.3 Legibilidad del entorno

Lo que hace que un agente se oriente rápido es lo mismo que ayudaría a un colega nuevo: estructura predecible, fronteras explícitas entre las partes del proyecto, y convenciones estándar de la disciplina. Si un recién llegado entendería de un vistazo qué hay en cada carpeta y qué hace cada archivo, el agente también. Las estructuras crípticas y los nombres internos cuestan tiempo y recursos a cada sesión.

Una prueba que no cuesta nada. El cold-start test no requiere herramientas: basta abrir una sesión limpia y observar. Conviene hacerlo periódicamente, porque la legibilidad de un proyecto se degrada sola a medida que crece.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Ejecutar un cold-start test sobre un proyecto y diagnosticar su resultado.
- Interpretar un cold-start lento como exceso o mala orientación de la documentación, y uno fallido como conocimiento no codificado.
- Identificar conocimiento tácito relevante y convertirlo en artefactos explícitos del proyecto.
- Evaluar la legibilidad de un entorno con el criterio del «colega nuevo» y corregir estructuras o nombres crípticos.

Errores y antipatrones frecuentes

Conocimiento en la cabeza. Confiar en que «ya me acuerdo» en lugar de escribir las decisiones y los porqués en el proyecto.

Documentar de más. Acumular documentación que el agente no necesita, ralentizando su orientación en lugar de acelerarla.

Estructura creativa. Nombres internos y carpetas crípticas que obligan al agente a gastar contexto averiguando qué es cada cosa.

Para profundizar

· Lopopolo (OpenAI) · Harness engineering

· Anthropic · Effective harnesses for long-running agents

BLOQUE C · PRINCIPIOS QUE GOBIERNAN EL ARNÉS

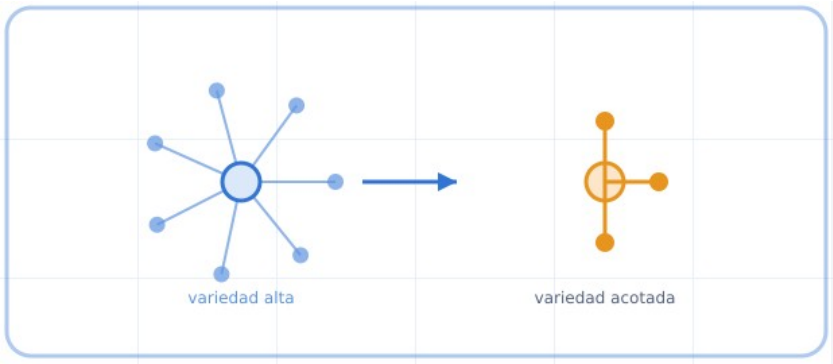
Capítulo 11. Ley de Ashby y reducción de variedad

EN CONSOLIDACIÓN

Un regulador necesita tanta variedad como el sistema que gobierna. Como igualar la del modelo es imposible, la salida es reducir la variedad del sistema.

Introducción

Hay un principio de la cibernética que explica, a nivel profundo, por qué el arnés se diseña como se diseña. La Ley de la Variedad Requisita, formulada por W. Ross Ashby en 1956, dice que un regulador debe tener al menos tanta variedad —tantos estados posibles— como el sistema que pretende gobernar. Solo la variedad absorbe variedad.



En lugar de igualar la variedad casi infinita del modelo, se reduce la del sistema atándolo a una topología concreta.

11.1 El problema aplicado al arnés

Un modelo de lenguaje puede producir casi cualquier cosa: su variedad es enorme. Según Ashby, para regularlo del todo tu arnés tendría que tener una variedad comparable, lo que es inviable. Si intentas anticipar y controlar cada posible salida con una regla, acabas con un arnés inflado e inmanejable que el modelo, además, deja de leer con atención.

11.2 La solución: invertir el principio

La salida práctica no es aumentar la variedad del regulador, sino reducir la del sistema. Si comprometes al agente con una topología concreta —una arquitectura conocida, un stack fijo, una plantilla cerrada de informe, un proceso metodológico definido, un corpus de fuentes acotado—, el espacio de cosas que puede producir se encoge, y entonces un arnés comprehensivo se vuelve posible. Restringir no empobrece el resultado: lo hace gobernable.

De aquí salen las plantillas de arnés. Esta es la justificación teórica de las plantillas de arnés del capítulo 13 y de toda la disciplina de acotación. Atar el agente a una topología no es una limitación que se sufre, sino la condición que hace posible regularlo bien.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Enunciar la Ley de la Variedad Requisita y aplicarla a la relación entre arnés y modelo.
- Explicar por qué igualar la variedad del modelo con reglas es inviable y conduce a arneses inflados.

- Reducir la variedad de un proyecto comprometiéndolo con una topología concreta (arquitectura, plantilla, proceso, corpus).
- Justificar la acotación como condición de gobernabilidad, no como pérdida de calidad.

Errores y antipatrones frecuentes

Regular con más reglas. Intentar cubrir cada salida posible del modelo añadiendo reglas, hasta inflar el arnés y perder señal.

Dejar el sistema abierto. No comprometer al agente con ninguna topología y esperar resultados consistentes de un espacio de salida ilimitado.

Ver la restricción como defecto. Resistirse a acotar por miedo a perder flexibilidad, cuando es justo la acotación lo que permite el control.

Para profundizar

· Böckeler · Harness engineering (martinfowler.com)

· Ley de la variedad requisita (Ashby)

BLOQUE D · PREPARAR EL ENTORNO

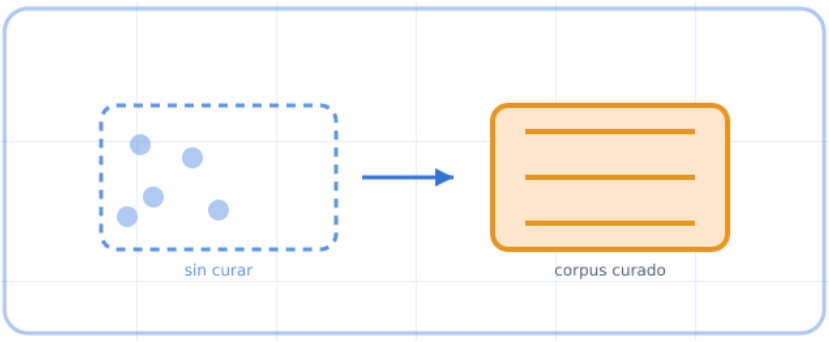
Capítulo 12. Pre-flight: espacio, confidencialidad y materiales

ESTABLECIDO

Antes de tocar ninguna herramienta: un espacio de trabajo dedicado, unas reglas mínimas de confidencialidad y unos materiales acotados y curados.

Introducción

Buena parte de los fallos sustantivos del trabajo agéntico se gestan antes de la primera instrucción, en un entorno mal preparado. Hay tres cosas que conviene tener listas con independencia de la herramienta que se use; son los errores más baratos de prevenir y los más caros de arrastrar.



Un corpus acotado y curado produce trabajo verificable; dejar al agente «buscar lo que encuentre» produce trabajo plausible pero no verificable.

12.1 Un espacio de trabajo, no una conversación suelta

El agente necesita un lugar donde viva el proyecto. Tanto en código como en texto, no es lo mismo una conversación dispersa que un proyecto con vida propia. Para cada proyecto conviene tener un contenedor dedicado en la plataforma (un Project, un repositorio, un workspace), una carpeta correspondiente para entradas y salidas, y un sistema de versionado de los entregables: Git si se trabaja con código, un sistema con historial —documentos compartidos con versiones, carpetas con fechas— si se trabaja con texto. Una conversación suelta no es un proyecto; es una mesa de trabajo provisional.

12.2 Confidencialidad y secretos

Un agente con permisos para ejecutar acciones tiene, en el peor caso, la misma capacidad que tu cuenta de usuario en los sistemas a los que accede. Reglas mínimas universales:

- Nunca pegar credenciales, claves ni secretos en el chat de un agente: servicios conectados, hooks o subagentes podrían leerlos.
- Nunca subir datos altamente confidenciales a versiones de consumidor de productos generales; usar planes empresariales o despliegues controlados vía API.
- Anonimizar datos antes de subirlos cuando se pueda hacer sin perder utilidad.
- Documentar en cada proyecto su nivel de confidencialidad y atenerse a las restricciones correspondientes.
- Preferir la autenticación gestionada por la plataforma o la CLI antes que pegar secretos en archivos del proyecto.

Verifica las políticas vigentes. Las condiciones de uso de datos de los fabricantes cambian. Antes de subir información sensible, y muy especialmente si manejas datos regulados, comprueba las políticas vigentes de la herramienta que uses.

12.3 Los materiales: el cimiento del trabajo

En programación los materiales primos son el código y las librerías; en investigación o análisis, fuentes, datos y documentos; en escritura profesional, briefs, transcripciones y referencias. Antes de empezar, acota y cura: reúne un corpus de referencia confiable y ponlo al alcance del proyecto, documenta qué materiales son aceptables y cuáles no, y lleva un registro desde el principio de dependencias, bibliografía y decisiones. Un agente con materiales curados produce trabajo verificable; uno que «se va a buscar lo que necesite» produce trabajo plausible pero no verificable.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Montar un espacio de trabajo dedicado con su carpeta y su sistema de versionado, en lugar de operar en una conversación suelta.
- Aplicar las reglas mínimas de confidencialidad sobre secretos y datos sensibles antes de subir nada a un agente.
- Documentar el nivel de confidencialidad de un proyecto y ajustar el entorno a ese nivel.
- Curar y acotar el corpus de materiales de un proyecto y dejar registro de qué es admisible.

Errores y antipatrones frecuentes

Trabajar en una conversación suelta. Tratar un proyecto serio como un chat provisional, sin contenedor ni versionado.

Secretos en el chat. Pegar credenciales o datos sensibles confiando en que «se quedan locales», ignorando hooks y servicios conectados.

Corpus abierto. Dejar que el agente busque libremente las fuentes, obteniendo resultados plausibles pero no verificables.

Para profundizar

· Lopopolo (OpenAI) · Harness engineering

· Anthropic · Effective context engineering

BLOQUE D · PREPARAR EL ENTORNO

Capítulo 13. El arnés mínimo viable y AGENTS.md

ESTABLECIDO

Cinco piezas bastan para estar haciendo Harness Engineering: instrucciones, estado, verificación, alcance y ciclo de vida.

Introducción

No hace falta un arnés sofisticado para empezar a trabajar bien. Un arnés externo mínimo viable —el cimiento sobre el que añadir sofisticación después— tiene cinco piezas. Si están las cinco, ya se hace Harness Engineering; lo demás amplifica, pero no sustituye este cimiento.

Las cinco piezas del arnés mínimo viable; con ellas presentes, ya se está haciendo Harness Engineering.

13.1 Las cinco piezas

Cada pieza cumple una función y se materializa en artefactos reconocibles:

1. Instrucciones: la guía maestra que el agente lee al empezar (archivo de instrucciones del proyecto, AGENTS.md, manual del proyecto).
2. Estado: la memoria persistente entre sesiones (lista de entregables con su estado, registro de decisiones, registro de progreso, sistema de versiones).
3. Verificación: los sensores que confirman la corrección (linter, tests, type checker, fact-checker, listas de comprobación, smoke tests).
4. Alcance: las reglas que restringen qué hace el agente en cada sesión (políticas declaradas, WIP = 1, pass-state gating).
5. Ciclo de vida: bootstrap, persistencia y limpieza del entorno (scripts de arranque, plantillas, rutinas de cierre).

13.2 El estándar AGENTS.md

El archivo de instrucciones se ha estandarizado en buena parte del ecosistema como AGENTS.md, un formato abierto en Markdown que actúa como «un README para agentes»: un lugar predecible donde poner el contexto y las reglas del proyecto. Surgió de un esfuerzo conjunto de varios fabricantes y hoy está mantenido por la Agentic AI Foundation bajo la Linux Foundation, con adopción en decenas de miles de proyectos. Su valor es la convergencia: se escribe una vez y lo leen múltiples herramientas.

El contrato de bootstrap. Dentro del ciclo de vida, el contrato de bootstrap es el conjunto de pasos —idealmente automatizables— que pone el entorno en estado conocido. En código suele ser un script idempotente que verifica dependencias, prepara el entorno y lanza una comprobación de

sanidad; en trabajo de conocimiento, una rutina documentada de «leer brief, leer entregables, leer progreso, confirmar contexto». Su existencia es lo que hace posible el cold-start test.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Enumerar las cinco piezas del arnés mínimo viable y la función de cada una.
- Identificar, en un proyecto propio, qué piezas están presentes y cuáles faltan.
- Redactar un archivo de instrucciones en formato AGENTS.md con las reglas globales del proyecto.
- Definir un contrato de bootstrap, como script o como rutina documentada, que ponga el entorno en estado conocido.

Errores y antipatrones frecuentes

Solo instrucciones. Escribir un buen archivo de instrucciones pero olvidar el estado persistente, la verificación o el ciclo de vida.

Sin contrato de bootstrap. No tener una forma reproducible de poner el entorno en estado conocido, lo que rompe el cold-start.

Reinventar el formato. Crear un formato propio de instrucciones cuando el estándar abierto ya lo leen las herramientas que usas.

Para profundizar

· AGENTS.md · estándar abierto

· Lopopolo (OpenAI) · Harness engineering

BLOQUE D · PREPARAR EL ENTORNO

Capítulo 14. Initializer + working agent: memoria en archivos

EN CONSOLIDACIÓN

El patrón canónico para tareas largas: un agente inicializa el entorno una vez; otro avanza sesión tras sesión leyendo y actualizando los archivos del proyecto.

Introducción

Las tareas que no caben en una sola ventana de contexto plantean un problema concreto: cada sesión empieza sin memoria de lo anterior. Anthropic documentó en noviembre de 2025 una solución que se ha convertido en el patrón canónico para tareas largas, directamente aplicable a programación, investigación, análisis y cualquier trabajo del conocimiento.

Un initializer prepara el entorno una vez; un working agent avanza sesión tras sesión; la memoria vive en los archivos.

14.1 Los dos roles

El initializer agent se ejecuta una sola vez: lee el brief, lo expande en una lista estructurada de entregables —cada uno con estado pendiente o hecho y criterios de aceptación claros—, monta el espacio de trabajo (estructura de archivos, plantillas, scripts de arranque, fuentes iniciales) y prepara las rutinas de verificación. El working agent se despierta una y otra vez; cada sesión ejecuta el bootstrap o se refundamenta leyendo los archivos, consulta el registro de progreso, elige una entrada pendiente, la produce y la verifica contra sus criterios, actualiza el registro de progreso con notas para el próximo turno, y cierra.

14.2 La clave: la memoria vive en los archivos

Lo esencial del patrón es que la memoria persistente no vive en la conversación ni en el contexto del modelo, sino en los archivos del proyecto. Cada sesión nueva se refundamenta leyéndolos. Es exactamente lo que haría un profesional humano al volver el lunes tras un fin de semana sin ver el proyecto. Esto conecta con WIP = 1 (capítulo 8) y con el cold-start test (capítulo 10): la refundamentación barata es lo que permite cerrar y abrir sesiones sin coste.

Patrones y repositorio de referencia. Anthropic ha publicado, además de los artículos fundacionales, un repositorio con estos patrones implementados como hooks y subagentes reutilizables, incluyendo un evaluador en contexto fresco y criterios que parten de «no cumplido». Es un buen punto de partida para no construir desde cero.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Describir los roles de initializer y working agent y qué hace cada uno en su momento.
- Expandir un brief en una lista estructurada de entregables con estado y criterios de aceptación.
- Diseñar el ciclo de una sesión: refundamentar, leer progreso, elegir una tarea, producir, verificar, actualizar progreso, cerrar.
- Hacer que la memoria persistente viva en los archivos del proyecto y no en la conversación.

Errores y antipatrones frecuentes

Memoria en la conversación. Confiar el estado del proyecto al hilo de chat, que desaparece al abrir una sesión nueva.

Sin lista de entregables. Trabajar sin una lista estructurada con criterios de aceptación, lo que impide saber qué falta y cuándo algo está hecho.

No actualizar el progreso. Cerrar una sesión sin dejar notas para la siguiente, obligando a cada turno a reconstruir el contexto desde cero.

Para profundizar

· Anthropic · Effective harnesses for long-running agents

· Anthropic · cwc-long-running-agents (GitHub)

BLOQUE E · PATRONES DE FRONTERA

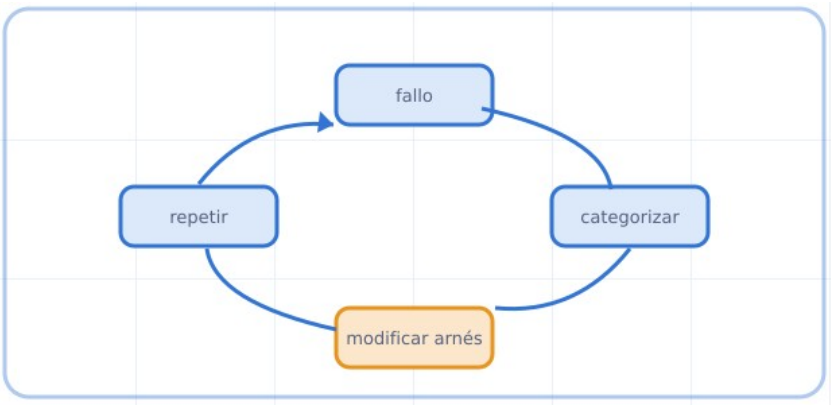
Capítulo 15. El bucle de dirección y el journal del arnés

EN CONSOLIDACIÓN

Cuando algo falla, se itera el arnés, no se regaña al modelo. El modelo no aprende del regaño; sí cambia si cambian sus instrucciones, señales y materiales.

Introducción

Böckeler lo llama steering loop. Es la práctica humana de iterar el arnés cuando algo falla, en lugar de iterar el prompt o regañar al modelo. Es el mecanismo que convierte el uso de agentes en una disciplina acumulativa en vez de una sucesión de frustraciones.



El bucle: ante un fallo, se categoriza, se modifica el arnés y se repite; regañar al modelo no cambia su comportamiento.

15.1 El procedimiento

Cuando un agente comete un error:

6. No le riñas al modelo: es ruido en su contexto y no resuelve nada estructural.
7. Categoriza el error: ¿falta una guía (no sabía la convención)?, ¿faltaba un sensor (no detectó que estaba mal)?, ¿falta acotar el alcance (intentó hacer demasiado)?, ¿falta legibilidad del entorno?
8. Modifica el arnés en consecuencia: a las instrucciones, al glosario o la plantilla; con un linter, un test o una skill; con más WIP = 1; con mejor estructura.
9. Repite la tarea y verifica que el cambio funciona.
10. Si vuelve a fallar el mismo tipo de error, el cambio no era el correcto: itera.

15.2 Por qué funciona

El modelo no aprende de tus regaños: no actualiza sus pesos. Pero sí cambia su comportamiento si cambian las instrucciones que lee, los errores que ve cuando se autocorrige y las herramientas y materiales a los que tiene acceso. Esas son las cosas que tú controlas, y cambiarlas es el único punto de palanca real que tienes. Un ejemplo: si el agente mete una cifra sin fuente y, al pedírsela, la inventa, la mala respuesta es «no inventes cifras»; la buena es añadir una regla de citación obligatoria y una skill que escanee el texto buscando cifras sin fuente adyacente, y pasarla como sensor antes de cada cierre.

15.3 El journal del arnés

Para que el bucle sea acumulativo conviene registrar cada cambio que haces al arnés y su resultado, en un documento propio. Cuando aparezca una técnica nueva en la industria, la contrastas con tu experiencia documentada. El journal es lo que distingue mejorar el arnés de parchearlo: convierte cada fallo en aprendizaje permanente del sistema, no del individuo.

La regla de oro del bucle. Ante un fallo, la pregunta nunca es «¿por qué el agente es tan torpe?», sino «¿qué capacidad falta en el entorno, y cómo la hago a la vez legible y verificable para el agente?». Esa reformulación es el corazón de la disciplina.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Ejecutar el procedimiento del bucle de dirección ante un fallo concreto, sin regañar al modelo.
- Categorizar un error como falta de guía, de sensor, de alcance o de legibilidad, y elegir la corrección adecuada.
- Convertir un regaño improductivo en un cambio estructural del arnés (regla, sensor, acotación).
- Mantener un journal del arnés que registre cambios y resultados para iterar de forma acumulativa.

Errores y antipatrones frecuentes

Regañar al modelo. Repetir «ya te lo dije» en el chat, gastando contexto sin cambiar nada estructural.

Cambiar sin categorizar. Tocar el arnés a ciegas sin diagnosticar si faltaba guía, sensor, alcance o legibilidad.

No registrar. Iterar el arnés sin journal, perdiendo el aprendizaje y repitiendo correcciones ya probadas.

Para profundizar

· Böckeler · Harness engineering (martinfowler.com)

· Anthropic · Scaling Managed Agents

BLOQUE E · PATRONES DE FRONTERA

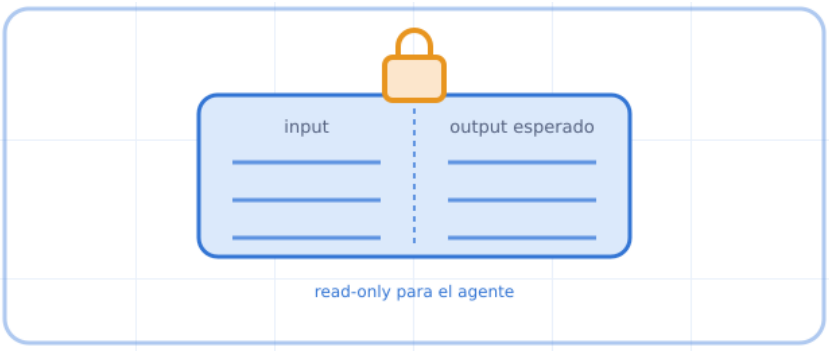
Capítulo 16. Behaviour harness, approved fixtures y revisores adversariales

EMERGENTE

Verificar que un entregable realmente hace lo que debía —no solo que está bien formado— es el frente abierto del campo. Hay patrones prometedores, no una solución general.

Introducción

El behaviour harness —cómo verificar que el entregable hace lo que tenía que hacer— sigue siendo la frontera abierta de la disciplina. Es a la vez el problema sin resolver y el de mayor consecuencia, porque los errores de sustancia son más difíciles de detectar que los de superficie.



Pares input/output marcados como de solo lectura: el agente debe satisfacerlos, no puede reescribirlos para «pasar».

16.1 El problema circular

Un entregable bien construido, bien estructurado y sin errores de superficie puede aun así no responder la pregunta que se planteaba, responderla en un nivel inútil de generalidad o resolver algo que no era el problema. A esto se suma un problema circular: las verificaciones que genera el propio agente tienden a validar lo que el agente acaba de producir. Si entendió mal el requisito, su verificación valida el malentendido.

16.2 Patrones prometedores

No hay solución general, pero sí direcciones que funcionan:

- Approved fixtures: el humano o un experto produce pares (input, output esperado) o (pregunta, respuesta mínima aceptable), marcados como de solo lectura. El agente debe satisfacerlos y no puede modificarlos; si cambia el output esperado para «pasar», rompe el contrato y un hook o un sensor lo detecta.
- Property-based testing: en lugar de casos puntuales, propiedades invariantes que deben cumplirse para cualquier input válido. Difícil de «engañar» con un property-test bien escrito.
- Revisores adversariales: un agente cuyo único trabajo es objetar, instruido para producir el peor memorando de objeciones que un par crítico podría escribir, sin suavizarse.
- El test del lector escéptico: identificar los tres lectores más sofisticados posibles y verificar que el entregable anticipa, o no es vulnerable a, sus objeciones.
- LLM como juez en contexto fresco: un evaluador que no vio cómo se produjo el entregable y le aplica criterios de aceptación claros, siempre como segunda línea tras los controles computacionales.

16.3 La implicación sobre la autonomía

La consecuencia práctica vuelve a aparecer aquí, ahora como principio de diseño: cuanto más crítica sea la corrección sustantiva, menos autonomía debe darse al agente y más rol humano debe haber. Los patrones de este capítulo reducen el riesgo, no lo eliminan, y se redefinen casi cada mes; conviene seguirlos sin adoptarlos por moda.

Por qué está en EMERGENTE. A diferencia del resto del arnés, aquí no hay consenso ni herramientas maduras. Los approved fixtures y los revisores adversariales son prometedores pero recientes; este es el capítulo cuyo contenido más probablemente cambiará en la próxima revisión del mapa.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Explicar la brecha de comportamiento y el problema circular de las verificaciones autogeneradas por el agente.
- Definir un conjunto de approved fixtures de solo lectura para un entregable concreto.
- Instruir un revisor adversarial y aplicar el test del lector escéptico a un entregable importante.
- Situar el LLM como juez como segunda línea de verificación, en contexto fresco, y no como única defensa.

Errores y antipatrones frecuentes

Verificación circular. Aceptar las comprobaciones que el propio agente generó, que validan su eventual malentendido del requisito.

Fixtures editables. Permitir que el agente modifique los pares esperados para hacerlos pasar, rompiendo el contrato sin que nadie lo note.

Toda la fe en el juez inferencial. Descansar la verificación de comportamiento en un único LLM evaluador, caro y falible, sin primera línea computacional.

Para profundizar

· Anthropic · Harness Design for Long-Running App Development

· Böckeler · Maintainability sensors for coding agents

BLOQUE E · PATRONES DE FRONTERA

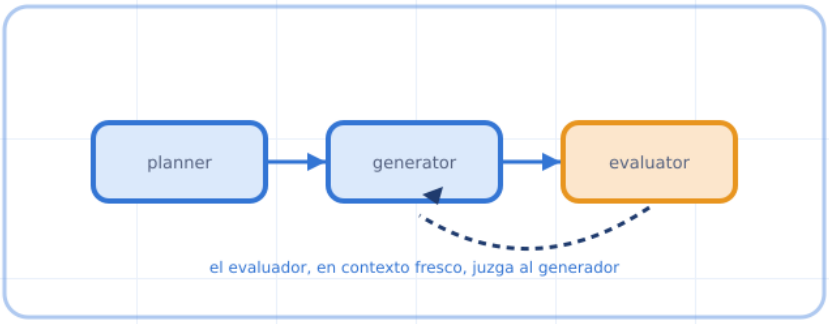
Capítulo 17. Arquitecturas multiagente y limpieza agéntica

EMERGENTE

La frontera de la orquestación: arquitecturas de varios roles que se vigilan entre sí, y agentes en segundo plano que limpian la deriva del proyecto.

Introducción

Cuando las tareas se vuelven largas y autónomas, el arnés evoluciona de un agente con sensores a varios agentes con papeles distintos que se vigilan entre sí, y a procesos de mantenimiento que corren solos. Es terreno potente pero volátil: las arquitecturas concretas cambian deprisa.



Una arquitectura de tres roles donde un evaluador, en contexto fresco, juzga el trabajo del generador en ciclos sucesivos.

17.1 Arquitecturas de varios roles

Anthropic ha documentado arquitecturas de tres roles —planner, generator y evaluator— inspiradas en las redes generativas adversariales: un evaluador con contexto fresco y sin permisos de escritura juzga el trabajo del generador en ciclos sucesivos, y cada ciclo produce un resultado más refinado. La separación de roles tiene una virtud clave: el evaluador no vio cómo se produjo el trabajo, así que no hereda los sesgos del generador.

17.2 Brain / Hands / Session

En una línea complementaria, Anthropic describe la separación Brain/Hands/Session: el Brain es el modelo más el bucle del arnés que lo invoca; las Hands son entornos aislados y efímeros donde se ejecutan las herramientas; la Session es un registro append-only de cada pensamiento, llamada y observación. La motivación es que cada componente del arnés codifica una asunción sobre lo que el modelo no puede hacer solo; al desacoplarlos, cuando una asunción envejece —porque el modelo nuevo ya puede hacer algo que antes requería una pieza— se cambia esa pieza sin rehacer todo el sistema.

17.3 Garbage collection agéntica

Todo proyecto acumula entropía: código muerto, dependencias huérfanas, fuentes que ya no se citan, glosario sin podar, decisiones revertidas sin anotar. La limpieza agéntica usa agentes en segundo plano con misiones específicas («encuentra material sin referencias y archívalo», «detecta inconsistencias terminológicas entre secciones») para corregirla de forma continua. Las reglas para que no se descontrole: el humano aprueba cada eliminación importante, se archiva antes de borrar, y cada limpieza deja una entrada en la bitácora del proyecto.

Pocos y bien definidos. La tentación de crear un subagente o una skill por cada problema lleva a quince que se solapan y disparan el coste. Si tienes más de un puñado activos en un proyecto, probablemente puedes consolidar. Son un martillo: no todo es clavo.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Describir una arquitectura de varios roles (planner/generator/evaluator) y la ventaja del evaluador en contexto fresco.
- Explicar la separación Brain/Hands/Session y por qué desacoplar evita rehacer todo el sistema cuando el modelo mejora.
- Diseñar una rutina de limpieza agéntica con sus salvaguardas: aprobación humana, archivar antes de borrar, bitácora.
- Mantener un número reducido de subagentes y skills, consolidando cuando se solapan.

Errores y antipatrones frecuentes

Acoplar todo el sistema. Mezclar modelo, bucle, sandboxes y registro, de modo que cualquier mejora del modelo obliga a rehacerlo entero.

Limpieza sin salvaguardas. Dejar que un agente elimine sin aprobación ni archivado, perdiendo material de forma irreversible.

Carrera de subagentes. Crear un nuevo subagente o skill ante cada problema, hasta acumular una maraña que nadie gobierna.

Para profundizar

· Anthropic · Harness Design for Long-Running App Development

· Anthropic · Scaling Managed Agents

BLOQUE F · CRITERIO PROFESIONAL Y ROL HUMANO

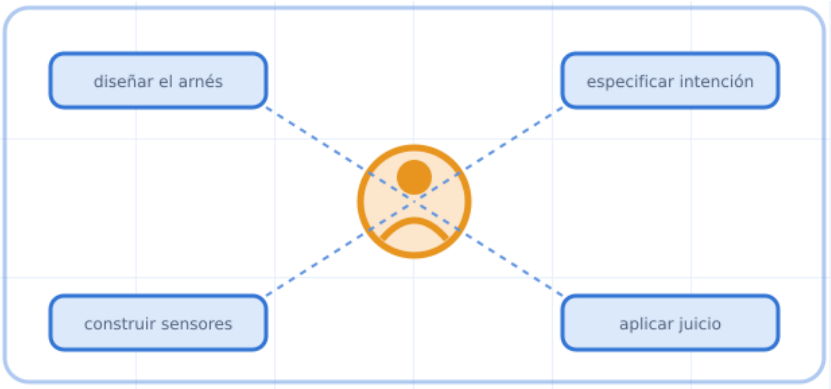
Capítulo 18. Medir el arnés, evitar antipatrones y sostener el criterio

EN CONSOLIDACIÓN

El arnés bien diseñado no elimina al humano: lo concentra donde más vale. Medir, reconocer antipatrones, saber cuándo no usar un agente y sostener el juicio.

Introducción

Este capítulo cierra la guía reuniendo la capa que protege la validez de todo lo anterior. Un arnés impecable puede, aun así, firmar trabajo que nadie ha pensado de verdad si falta medición, si se repiten antipatrones conocidos o si el humano abdica del criterio. Las cuatro secciones que siguen son las cuatro caras de esa capa.



El humano deja de escribir cada línea para diseñar el entorno, especificar la intención, construir los bucles de feedback y aplicar juicio donde importa.

18.1 Medir el arnés

Si no observas, no mejoras. No hace falta un sistema sofisticado: una hoja de cálculo con fecha, tarea, duración, iteraciones, bloqueantes y objeciones basta para que, tras veinte o treinta entradas, los patrones se vean. Los indicadores con mayor valor:

- Rebuild cost: tiempo desde una sesión vacía hasta trabajo productivo. Es el KPI norte; por debajo de unos pocos minutos, el arnés es de élite.
- Tasa de cierre a la primera: porcentaje de tareas que cierran sin más de una iteración mayor.
- Tasa de objeciones del destinatario: porcentaje de entregables que reciben objeciones sustantivas externas; debería estar al mínimo.
- Cobertura de sensores: porcentaje de cierres que han pasado por todos los sensores definidos; el objetivo es la totalidad.
- Latencia de mejora del arnés: tiempo desde detectar un patrón de fallo hasta cerrarlo en el arnés; idealmente menos de una sesión.

Conviene revisar estos indicadores periódicamente en proyectos largos; si alguno empeora, identifica qué cambió.

18.2 Antipatrones frecuentes

Los modos de fallo se repiten entre disciplinas. Reconocerlos antes de tiempo evita erosionar la fiabilidad:

- Instrucciones sobre-especificadas: el archivo crece sin parar, el modelo deja de leerlo con atención y las reglas críticas se pierden en el ruido. Antídoto: poda agresiva; si una regla no se ha violado en varias sesiones o es enforzable mecánicamente, conviértela en sensor y bórrala.
- Exploración infinita: «investiga X» sin acotar agota el contexto. Antídoto: acotar explícitamente fuentes y alcance.
- Confianza prematura: aceptar el «hecho» sin verificar. Antídoto: pass-state gating estricto.
- Bypass por desesperación: activar un modo permisivo para «que termine ya», con riesgo de algo irreversible. Antídoto: bypass solo en entornos aislados; si te tienta hacerlo en tu máquina principal, vete a dormir.
- Carrera de subagentes: crear uno nuevo ante cada problema hasta acumular una maraña costosa. Antídoto: pocos y bien definidos.
- Fe en el revisor inferencial: descansar toda la verificación en otro modelo. Antídoto: que sea segunda línea, no única.
- Dependencia oculta de una sola fuente o herramienta. Antídoto: una regla de diversificación y un sensor que reporte la distribución de fuentes al cierre.

18.3 Cuándo NO usar un agente

Parte del criterio profesional es saber cuándo la respuesta correcta es cerrar el chat y trabajar tú:

- Tareas cortas cuyo contexto solo está en tu cabeza: explicárselo al agente cuesta más que hacerlo.
- Decisiones críticas con responsabilidad organizacional: el agente no carga con la responsabilidad; tú sí, y tu juicio es lo que se paga.
- Cuando el cliente paga por tu juicio, no por tu output: acuerda con él dónde puede y no puede entrar la IA.
- Refactors o cambios profundos con dependencias sutiles: el agente rompe invariantes que no ve; mejor humano y agente como pareja.
- Cuando la confidencialidad es máxima: hay zonas donde el riesgo no se justifica ni con planes empresariales.
- Cuando tú no sabes la respuesta: sin una hipótesis informada propia, el agente alucinará algo plausible y tú lo firmarás.
- Cuando el sistema está roto y no sabes por qué: el agente alucinará causas plausibles que te despistan; déjate guiar por el debug humano.

18.4 El rol del humano

El arnés bien diseñado no elimina al humano: lo concentra donde más vale. El trabajo profesional ya no es escribir cada función o cada párrafo; es diseñar el entorno (el arnés), especificar la intención (el brief, los criterios de aceptación, las preguntas que el entregable debe responder, las invariantes que no se pueden violar), construir los bucles de feedback (sensores, verificaciones, revisiones críticas), aplicar juicio donde la máquina no debe decidir sola (la tesis, la recomendación final, la decisión arquitectónica, la matización legal o política, qué entra y qué no) e iterar el arnés cuando algo no funciona.

Más exigente, no menos. Esto es trabajo intelectual real, exigente de otra manera y más apalancado: un buen arnés produce semanas de output de calidad mientras tú piensas en lo que de

verdad merece tu pensamiento. Y conviene la humildad técnica: las verdades de este campo tienen vida corta, así que la disposición a desaprender vale más que cualquier certeza adquirida.

Lo que debes saber hacer

Al dominar este capítulo, un profesional es capaz de:

- Definir y seguir los KPIs del arnés —rebuild cost, cierre a la primera, objeciones, cobertura de sensores, latencia de mejora— con un registro simple.
- Reconocer los antipatrones frecuentes en un proyecto propio y aplicar el antídoto correspondiente.
- Decidir, ante una tarea concreta, si procede usar un agente o trabajarla a mano, según responsabilidad, confidencialidad y conocimiento propio.
- Delimitar el rol del humano como diseñador del arnés, especificador de intención, constructor de sensores y responsable último del criterio sobre el entregable.

Errores y antipatrones frecuentes

No medir. Trabajar sin ningún indicador, de modo que el arnés no mejora porque nadie sabe si empeora.

Abdicar del criterio. Aceptar la salida del agente en decisiones donde la responsabilidad y el juicio son intransferibles.

Usar el agente donde no toca. Recurrir a un agente para tareas que exigen contexto tácito, confidencialidad máxima o una hipótesis propia que no se tiene.

Para profundizar

· Böckeler · Harness engineering (martinfowler.com)

· Mollick · One Useful Thing