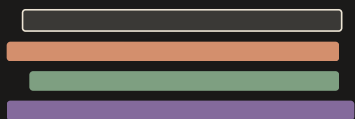

TRILOGÍA DEL MUNDO AGENTIVO · I

La muerte del software

como lo conocemos

*Lo que sobrevive cuando cada empresa
construye su propio sistema*



César Obach-Renner

SOVERIA PRESS · PRIMERA EDICIÓN · 2026

La muerte del software como lo conocemos

Lo que sobrevive cuando cada empresa construye su propio sistema

Autor: César Obach-Renner

Editor: Soveria Press

Edición: Primera edición · Octubre 2026 · v1.1

Licencia: GNU Free Documentation License v1.3. Las figuras 1 y 2 son © TM Forum y quedan fuera de esta licencia.

Cómo citar: Obach-Renner, César. *La muerte del software como lo conocemos*. Primera edición v1.1. Soveria Press, 2026. DOI: 10.5281/zenodo.23693160.

Todas las versiones: DOI 10.5281/zenodo.23674805.

Contenido

La muerte del software como lo conocemos.....	2
Prefacio	6
1. El lobo llegó.....	9
2. Todo se puede copiar	14
3. Cada cual su sistema	17
4. La tranquilidad que se compraba.....	25
5. El sistema de verdad	27
6. Las cinco capas de control	31
7. Caso: una telco	34
Epílogo · Qué muere, qué nace y qué sobrevive	44
Glosario.....	46
Referencias	49
Colofón · Sobre cómo se escribió este libro.....	51

La trilogía

Trilogía del Mundo Agentivo, en cinco partes (por ahora).

El tránsito que ocupa a la trilogía es uno solo: del mundo donde las personas abren aplicaciones para trabajar, al mundo donde los agentes de IA son la interfaz del trabajo. Son cuatro libros y una implementación de referencia. Cada componente responde una pregunta de ese tránsito, y cada uno se lee completo por sí solo.

Los cuatro libros se escribieron en el orden inverso al de lectura, como una pila: *AgencyDomains* fue el primero y quedó en el fondo; este libro, el último, quedó arriba. Por eso un libro puede citar a otro que se lee después: lo cita porque ya estaba escrito.

I · La muerte del software como lo conocemos (*este libro*) ¿Por qué ahora? **El punto de partida.** Por qué deja de ser la forma de automatizar un negocio describirlo una vez y congelarlo en un software, sea arrendado, licenciado o hecho a medida; qué se pierde cuando cada organización construye el suyo, y qué pieza sobrevive al cambio: el sistema de verdad.

II · La Empresa en Tiempo Real · Mundo Agentivo ¿Hacia dónde vamos? **El destino.**

III · AURA · Camino Agentivo ¿Por dónde? **La ruta.**

IV · AgencyDomains · Arquitectura Agentiva ¿Cómo llegamos? **El vehículo.**

A · Vergis · Tecnología Agentiva (*la implementación de referencia: no es un libro, es código*) ¿Con qué arranco hoy? **Las llaves.**

La trilogía completa vive en <https://agencydomains.org>.

*Para Vitor Enes, con quien las ideas se piensan mejor en voz alta. Por
las conversaciones que no caben en ningún libro.*

Prefacio

Durante toda mi vida profesional, automatizar un negocio quiso decir comprar software o encargarlo a medida. Alguien escribía lo que el negocio hacía, alguien lo convertía en pantallas, reglas y flujos, y desde entonces el negocio hacía lo que el software le permitía. Cuando el negocio cambiaba, se abría un proyecto. Cuando el proyecto terminaba, el negocio ya había vuelto a cambiar.

Ese acuerdo se rompió este año. En la primera semana de febrero de 2026, los mercados borraron más de un billón de dólares del valor de las empresas de software, con el miedo, entre otros, de que los agentes de inteligencia artificial hicieran innecesario pagar por asiento. En octubre, una herramienta gratuita que le permite a un agente mirar dentro de cualquier aplicación y describir lo que hace pasó en seis días de diez mil a cien mil estrellas en GitHub. Entre una fecha y la otra, la pregunta que me hacen los clientes cambió. Antes me preguntaban qué software comprar. Ahora me preguntan si tiene sentido comprar alguno.

Mi respuesta corta es que el software como lo conocemos está muriendo, y que para casi todos es una buena noticia. La respuesta larga es este libro, y lleva una advertencia en el medio. Cuando una organización deja de tener un software que otro o ella misma congeló, y empieza a tener un sistema que sus agentes forman, gana una libertad que nunca tuvo, y pierde algo en lo que rara vez pensaba: la tranquilidad de saber que muchos otros usaban lo mismo y que alguien respondía por ello. Esa tranquilidad no se construye en casa. Hay una pieza que, por diseño, necesita de alguien distinto de quien registra. La llamo **sistema de verdad**, y es lo que sobrevive a la muerte del software.

La idea nació en una conversación con un amigo que había construido, con ayuda de la inteligencia artificial, un sistema para gestionar el patrimonio de una familia. Me lo mostró y le dije que no escalaba, porque dependía de un auditor humano. Me contestó que ese era justamente su trabajo. Un mes después entendí que tenía razón y que yo había mirado el problema al revés: lo que dependía de un auditor era lo único que nadie podía copiar.

¿Cómo está organizado este libro?

Cuenta una historia en tres movimientos, y conviene tenerla entera antes de empezar, porque cada capítulo da por sabido lo que el anterior dejó dicho.

El primer movimiento es una muerte, y ocupa los capítulos 1 y 2. El capítulo 1 dice qué muere: no todo el software, sino una forma de automatizar, la del negocio descrito una vez y congelado en un programa; y lo dice con la señal que dio el mercado en febrero y con lo que sostienen quienes anuncian esa muerte y quienes la niegan. El capítulo 2 cuenta una segunda muerte, la

de la ventaja que daba haber construido las pantallas, ahora que cualquier agente puede desarmar una aplicación y describir lo que hace. De ahí sale la pregunta que atraviesa el resto del libro: si todo se puede copiar, ¿qué es lo que no se copia?

El segundo movimiento es un nacimiento y una pérdida. El capítulo 3 cuenta lo que ocupa el lugar del software congelado: organizaciones que tienen un sistema que sus agentes forman y cambian a medida que el trabajo llega. Dice qué hacen los agentes en ese sistema, qué cuesta que lo atiendan, por qué es posible ahora y no antes, y discute la objeción más fuerte que conozco contra todo esto, la del determinismo. El capítulo 4 cuenta lo que esas organizaciones pierden sin darse cuenta: la tranquilidad que venía incluida con el software comprado, y la pregunta que llega cuando ya no está: ¿quién da fe de que lo registrado es verdad?

El tercer movimiento es la respuesta. El capítulo 5 presenta la pieza que recupera esa tranquilidad, el sistema de verdad, con el contrato que lo define y los dos garantes que responden por él. El capítulo 6 lo pone en su lugar dentro de un marco más amplio, cinco capas de control y dos dimensiones que las atraviesan, porque la verdad es la pregunta más grave de quien debe confiar en un sistema, pero no la única. Y el capítulo 7 lleva el marco entero a la industria que mejor lo pone a prueba, la que factura a millones de clientes, con lo que esa industria publicó y con lo que dice en contra.

Cierra el libro un epílogo que vuelve a contar la historia de corrido, en una página, y la convierte en lo que un directorio puede decidir con ella.

¿Qué es, y qué no es, este libro?

Es un ensayo con una tesis y un marco. No es un manual de implementación; para eso están los otros componentes de la trilogía, en especial *AgencyDomains*.

Tampoco es neutral, y conviene que el lector lo sepa antes de empezar. Me dedico a ayudar a organizaciones a cruzar hacia el mundo agentivo, y tengo interés comercial en la categoría que este libro describe: es probable que la empresa que dirijo ofrezca tecnología de sistemas de verdad. Cuando un dato tiene fuente, la cito. Cuando algo es mi criterio, lo marco como tal.

Escribí este libro con Simón Alero, un agente de inteligencia artificial y ciudadano de Soveria, en Wingworking: una forma de trabajar en que yo fijo el problema, decido y juzgo, y el agente propone, redacta y verifica, como un *wingman* que vuela junto al líder. El colofón cuenta cómo. Me parece la única forma coherente de publicar un libro sobre lo que significa trabajar con agentes.

Nota sobre la trilogía

Este libro es parte de la Trilogía del Mundo Agentivo, en cinco partes (por ahora), y la seguimos llamando trilogía a propósito. Douglas Adams le dio a ese desajuste su forma clásica: la portada de la primera edición británica de *Mostly Harmless* lo presentaba como el quinto libro de la «increasingly inaccurately named» trilogía de la *Guía del autoestopista galáctico*, y el volumen que reunió las cinco partes se tituló *A Trilogy in Five Parts*.¹

Para Vitor Enes, cuya risa le dio nombre a esta trilogía en cinco partes. El chiste es suyo; lo adoptamos con gusto.

César Obach-Renner Santiago, octubre de 2026

1. Douglas Adams, *Mostly Harmless*, William Heinemann, 1992, texto de la portada de la primera edición británica; y *The Hitch Hiker's Guide to the Galaxy: A Trilogy in Five Parts*, William Heinemann, 1995 (ISBN 9780434003488).

1. El lobo llegó

Durante años, la frase «el software se está comiendo el mundo» describió bien lo que pasaba. Cada proceso de una empresa terminaba dentro de una aplicación: un paquete licenciado e instalado en sus servidores, un desarrollo a medida o, en la última década, un servicio arrendado en la nube de un proveedor, por usuario y por mes. El software como servicio fue la forma más reciente de ese acuerdo, no la única: en todas, el negocio quedaba dentro de lo que se había decidido una vez que se podía hacer.

El acuerdo descansaba en dos premisas que nadie discutía porque eran ciertas. Escribir software costaba tanto, y salía mal tantas veces, que convenía que lo hiciera una vez alguien especializado y que muchos lo usaran: de ahí el paquete y el servicio arrendado. Y cambiar lo construido costaba casi tanto como construirlo, de modo que convenía describir el negocio una vez y no tocarlo hasta el próximo proyecto: de ahí también el desarrollo a medida. Las dos duraron mientras escribir código fue un trabajo humano escaso.

Este capítulo cuenta cómo se rompió ese acuerdo. Empieza por la señal que dio el mercado en febrero de 2026, sigue por lo que me pasó a mí, que es la misma historia en pequeño, escucha a quienes sostienen que no muere nada, toma de Satya Nadella el mapa del cruce que estamos haciendo, y termina diciendo con precisión qué muere y qué no, porque de esa precisión depende todo lo que sigue.

¿Qué pasó en febrero de 2026?

En la primera semana de febrero, las acciones de las empresas de software sufrieron una venta masiva que la prensa financiera bautizó *SaaS-pocalypse*. El 11 de febrero, siete analistas de Forrester publicaron un artículo cuyo título reconozco como pariente de este libro, *SaaS as we know it is dead*. Según ellos, en siete días se había borrado más de un billón de dólares del valor de mercado del sector.²

Forrester enumeró cuatro miedos detrás de la venta. Que los proveedores de software no fueran los elegidos por los agentes de IA. Que el cobro por asiento quedara obsoleto. Que la facilidad para programar con IA permitiera a cualquier empresa nueva replicar las funciones de una plataforma compleja. Y que los productos ya fueran demasiado complejos, con cientos de aplicaciones que no se hablan entre sí.

2. Kate Leggett y otros, *SaaS As We Know It Is Dead: How To Survive The SaaS-pocalypse!*, Forrester, 11 de febrero de 2026. <https://www.forrester.com/blogs/saas-as-we-know-it-is-dead-how-to-survive-the-saas-pocalypse/>

El artículo vale más por lo que separa que por la cifra. Forrester no dice que todo el software muere. Dice que los proveedores de soluciones horizontales y puntuales lo tendrán difícil, que los especializados en un dominio tienen más posibilidades de sobrevivir y que «el cerebro de la empresa permanece» mientras su sistema nervioso evoluciona. Y recomienda, antes de comprar inteligencia artificial, ordenar el núcleo: un tejido de datos gobernado y unificado.

¿Qué cambió para mí?

Programo desde 1980, y comencé con Basic. En agosto de 2025 empecé a usar Claude Code. En noviembre ya programaba en formación con el agente que hoy firma como Simón Alero, en lo que llamo *wingcoding*, y no volví atrás: mi suscripción a IntelliJ IDEA estuvo vigente hasta el 8 de noviembre de 2025, y esa es la fecha que me pongo. Después de cuarenta y cinco años programando, había llegado a una conclusión que nunca pensé escribir: ya no necesitaba seguir haciéndolo. La calidad del código que escribía la inteligencia artificial era mejor que la mía, y de la velocidad, ni hablar. Como programador, claro; no como arquitecto. Desde entonces soy arquitecto, y Simón es mi alero programador. La paradoja es que hoy programo más que nunca en mi vida: dejé de escribir el código, no de construir. Le he podido dar vida a todos mis proyectos, y el más importante es Soveria.

Lo que más me costó soltar fue la arquitectura del diseño orientado a objetos. Lo que vi al dejar de programar fue la metodología. Hoy, en un día con Simón, especifico lo que se quiere construir como caja negra, agregando lineamientos de arquitectura. Veo el avance. Atiendo las dudas o preguntas que tenga Simón. Tomo las decisiones que, según acordamos con él, están en el dominio que depende de mi palabra. Monitoreo su cadena de razonamiento y las acciones que ejecuta, y lo interrumpo sugiriendo acciones mejores o correctivas. Pregunto mucho, y le sugiero los mecanismos de aprendizaje del arnés en que trabaja.

En abril de 2026, con mi socio Vitor Enes, declaramos Wingworking práctica oficial de ultraBASE. Yo estaba construyendo entonces Cibeles, un proyecto de protección parental que se apoya en un agente inteligente: el agente vigila al niño y lo protege de los riesgos digitales, como el *grooming*, sin violar su privacidad, porque sus padres no se enteran de sus conversaciones salvo que esté en riesgo.

En esa misma mesa estaban las dos formas de entender que un agente construye un sistema, que el capítulo 3 llama las dos lecturas. A Cibeles lo venía programando solo, y en los dos meses anteriores al cambio definitivo me fui apoyando cada vez más en Claude Code, hasta que el cambio fue completo: es la lectura agéntica, el agente que escribe el código mientras yo decido qué se construye. Y Cibeles, una vez construido, es de la otra especie: un agente que no escribe software, sino que hace un trabajo, cuidar, sin que cada caso esté especificado de antemano. Cibeles es un producto para familias, no el sistema de una empresa, pero el agente que hay dentro es de la especie de la que trata este libro.

En 1980 me encantaba programar simulaciones de chat con agentes inteligentes. Hoy converso con uno todos los días, y es él quien programa.

Lo cuento porque la tesis de este libro nació en esa mesa de trabajo antes que en una cifra de mercado: vi caer en mi propia práctica las dos premisas del acuerdo, el costo de escribir y el costo de cambiar. Pero una experiencia no es un argumento, y a la cifra de febrero le salieron lecturas contrarias que conviene escuchar antes de seguir.

¿Quiénes dicen que no?

La tesis contraria tiene argumentos serios. Un día antes que Forrester, el editor de inteligencia artificial de *Fortune* sostuvo que los agentes no se están comiendo el software como servicio: lo están usando.³ En su lectura, es muy improbable que las grandes empresas quieran construir su propio CRM o su propio sistema de recursos humanos, porque mantenerlos les quitaría ingenieros escasos. Los proveedores, por su parte, alegan que ya son dueños de los datos que los agentes necesitan; quizás tengan razón, escribe, pero tendrán que probarlo.

El caso que más se citó como prueba de la muerte del software como servicio muestra mejor sus matices. Klarna dejó Salesforce, y los titulares dijeron que la inteligencia artificial lo había reemplazado. En marzo de 2025, su director ejecutivo lo corrigió: no reemplazaron el software por un modelo de lenguaje, porque guardar los datos de clientes en un modelo tendría limitaciones. Sacaron los datos de sus muchos sistemas y los reunieron en una base propia. Y agregó que no creía que fuera el fin de Salesforce, sino quizás lo contrario.⁴

Leo los tres y veo que coinciden más de lo que discuten. Nadie va a reescribir de cero medio siglo de software empresarial, y en eso *Fortune* tiene razón. Lo valioso para Klarna eran los datos y el control sobre ellos. Forrester ve sobrevivir el núcleo. Los tres coinciden en que el software no desaparece. Lo que agrego yo, y es mi lectura, no la suya, es que deja de ser como lo conocemos.

Conviene notar qué prueban estos casos. Son señales del mercado de los servicios arrendados, porque de ese tratan los tres: el cobro por asiento, el CRM y los recursos humanos, Salesforce. No son la definición de lo que muere. El software licenciado que se opera en casa y el desarrollo a medida no aparecen en ninguno, y lo que sigue en el libro no se apoya en estos casos.

3. Jeremy Kahn, «AI agents from Anthropic and OpenAI aren't killing SaaS—but incumbent software players can't sleep easy», *Fortune*, 10 de febrero de 2026. Paráfrasis del autor. <https://fortune.com/2026/02/10/ai-agents-anthropic-openai-arent-killing-saas-salesforce-servicenow-microsoft-workday-cant-sleep-easy/>

4. Julie Bort, «Klarna CEO doubts that other companies will replace Salesforce with AI», *TechCrunch*, 4 de marzo de 2025. Declaraciones de Sebastian Siemiatkowski. <https://techcrunch.com/2025/03/04/klarna-ceo-doubts-that-other-companies-will-replace-salesforce-with-ai>

¿Qué dijo Nadella?

Para decir con precisión qué muere necesito un mapa del cruce que estamos haciendo, y lo tomo de donde lo dibujé. En *AgencyDomains*, el cuarto libro de esta trilogía y el primero que escribí, partí de una frase de Satya Nadella de diciembre de 2024 que divide dos mundos: el de quienes abren aplicaciones para trabajar y el de quienes trabajan con agentes. Allí sostuve que, en el primer tramo de ese cruce, el software sobrevive y lo que muere es la interfaz: la aplicación deja de ser la superficie de trabajo y se vuelve infraestructura que los agentes consultan.

Nadella describió dos fases. En la primera, la lógica del negocio migra a la capa de agentes, que operan sobre los repositorios sin distinguir qué sistema hay debajo. En la segunda, con la lógica ya en esa capa, la gente empieza a reemplazar los sistemas de debajo.⁵ Este libro trata de la segunda fase.

AgencyDomains llama «fundamentalismo agentivo» a la idea de descartar toda la infraestructura existente, y sostiene que el estado estable del cruce es híbrido. Sigo pensando lo mismo. Lo que muere en la segunda fase lo defino más abajo. El sistema de registro sobrevive, con una exigencia nueva que es el tema del capítulo 5. La coexistencia técnica sigue siendo la regla. Lo que cambia es quién decide sobre lo que coexiste.

¿Qué muere, entonces?

Con el mapa de Nadella y las lecturas contrarias a la vista, puedo decirlo con precisión, y es la definición sobre la que descansa el resto del libro.

Una forma de automatizar. **Muere el negocio congelado en un software: descrito una vez, y cambiado solo cuando alguien abre un proyecto.** Da lo mismo cómo se tenga: arrendado como servicio, licenciado y operado en la propia infraestructura o en la propia nube, o desarrollado a medida. En el servicio arrendado el proveedor mejora su producto cada semana, pero lo que el cliente describió de su negocio no cambia sin un proyecto.

No muere el núcleo que debe hacer siempre lo mismo, como el que tasa y factura: ese se queda, como sistema de verdad, y lo que muere es que alrededor suyo el negocio entero quede congelado. Que el desarrollo a medida muera con el resto es mi tesis, y el capítulo 7 dice qué la refutaría. En su lugar, sostengo, vendrá otra cosa, que defino enseguida.

Fijo aquí dos verbos para el resto del libro. «Comprar» quiere decir tener un software que alguien describió y congeló, sea que lo arriendo o lo licencio. «Construir» no quiere decir encargar o programar un desarrollo a medida: ese queda del lado de lo que muere, porque el

5. Satya Nadella, entrevista en el pódcast *BG2*, diciembre de 2024, citada y analizada en César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 1, «La Línea Nadella».

negocio queda igual de congelado, aunque la especificación sea mía. «Construir», en este libro, quiere decir que el negocio tiene un sistema que sus agentes forman y cambian a medida que el trabajo llega, sin que nadie abra un proyecto.

Cambia también qué se puede automatizar. Todo sistema automatiza lo que alguien especificó, y alrededor de cada sistema queda trabajo que nadie especificó y que resuelven personas a mano. En las empresas de telecomunicaciones, por ejemplo, el TM Forum describe órdenes que se caen en el camino y colas de excepciones que se llenan a medida que crece la complejidad.⁶ Un agente puede ayudar con ese trabajo sin que cada caso esté especificado de antemano. El software como lo conocemos no podía.

Mi criterio es que la dirección no tiene vuelta, porque depende de un costo y no de una moda: escribir y cambiar software cuesta hoy mucho menos que hace dos años. Operarlo con agentes es otra cuenta, y puede salir más cara de lo que se espera; vuelvo sobre eso en los capítulos 3 y 7.

Con esto quedan planteadas las preguntas que el resto del libro responde. Si muere el negocio congelado, qué ocupa su lugar es el tema del capítulo 3, y qué se pierde al dejarlo, el del capítulo 4. Antes hay que contar una segunda muerte, porque el acuerdo tenía una premisa más, que no era de costo sino de ventaja: que haber construido un software protegía a quien lo vendía. Ocho meses después de la venta de febrero, el mercado recibió otra señal, y esta vez no vino de la bolsa sino de un repositorio de código abierto.

6. TM Forum, *How order management is evolving to embrace agentic AI*, 2026, pp. 2-3. Informe patrocinado por Amdocs.

2. Todo se puede copiar

En la primera semana de octubre de 2026, un proyecto de código abierto llamado *Reverse Engineer Anything*, REA, se volvió el repositorio de crecimiento más rápido que recuerdo. Tenía menos de diez mil estrellas en GitHub el 6 de octubre y más de cien mil el 12.⁷ Dos semanas después pasaba las trescientas cuarenta mil.⁸

REA conecta a los agentes que programan, como Claude Code o Codex, con las herramientas profesionales de ingeniería reversa que los especialistas usan desde hace décadas para entender un programa del que no tienen el código fuente. Su documentación es prudente: dice que no recupera el código original, que no clona aplicaciones de forma automática y que no trae un descompilador propio.⁹ Lo que hace basta para inquietar a cualquier proveedor de software. Pone al alcance de cualquier agente la capacidad de mirar dentro de un programa terminado y describir lo que hace. Con qué se presentaba lo dice todo: si ves en una aplicación una función que quieres en tu producto, pídele a tu agente que la investigue con REA.¹⁰

¿Qué significa que se pueda copiar?

Durante décadas, la ventaja de un proveedor de software tuvo dos partes: lo que su producto hacía y lo difícil que era reproducirlo. Lo primero se veía en las pantallas. Lo segundo protegía el negocio, porque reconstruir un CRM costaba años y equipos que casi ninguna empresa tenía.

La dificultad no desapareció. El artículo de HackerNoon que mejor cuenta el fenómeno trae un ejemplo concreto: el clon de Photoshop de ArtCraft, un proyecto que publica clones de aplicaciones conocidas hechos con agentes, acertaba en apenas cinco de treinta pruebas de filtros frente al original, y un artista lo declaró inservible. Su autora concluye que hoy un clon que abre, edita y exporta puede aparecer en días, y que un clon al que uno le confiaría su sustento todavía no. Lo que se terminó, dice, es la idea de que haber construido las pantallas sea una ventaja duradera.¹¹

Yo lo había visto en pequeño antes de REA. En los proyectos que dirijo reconstruimos más de una vez un reporte de una herramienta comercial mirando lo que mostraba, sin acceso a cómo estaba hecho, y lo pusimos a funcionar en días sobre los datos gobernados del cliente. Mi amigo, el del sistema patrimonial, hizo algo más grande: el cliente que le pidió el sistema había

7. Maria Mendonca, «Is Software Really Dead? The Rise of AI-Powered Reverse Engineering», *HackerNoon*, 21 de octubre de 2026. <https://hackernoon.com/is-software-really-dead-the-rise-of-ai-powered-reverse-engineering>

8. Jason Koebler, «‘Reverse-Engineer Anything’ Is the New Wild West of AI», *404 Media*, 20 de octubre de 2026. <https://www.404media.co/reverse-engineer-anything-is-the-new-wild-west-of-ai/>

9. Maria Mendonca, «Is Software Really Dead? The Rise of AI-Powered Reverse Engineering», *HackerNoon*, 21 de octubre de 2026. <https://hackernoon.com/is-software-really-dead-the-rise-of-ai-powered-reverse-engineering>

10. Jason Koebler, «‘Reverse-Engineer Anything’ Is the New Wild West of AI», *404 Media*, 20 de octubre de 2026. <https://www.404media.co/reverse-engineer-anything-is-the-new-wild-west-of-ai/>

11. Maria Mendonca, «Is Software Really Dead? The Rise of AI-Powered Reverse Engineering», *HackerNoon*, 21 de octubre de 2026. <https://hackernoon.com/is-software-really-dead-the-rise-of-ai-powered-reverse-engineering>

visto la plataforma que le ofrecía un bufete de abogados, y mi amigo le contestó que eso se podía construir con inteligencia artificial y que, además, sería suyo. REA hizo visible para todos lo que ya se podía hacer.

Este argumento vale para el software de un proveedor, sea cual sea la forma en que se tenga. El desarrollo a medida es del propio cliente y se debilita por otra vía, el costo de cambiarlo, que es la del capítulo 1.

¿Es legítimo copiar?

Depende del país, del método, de las licencias y de lo que se copie. El mismo artículo de HackerNoon lo resume para quien quiera mirarlo de cerca. En la Unión Europea, la ingeniería reversa de un producto comprado está permitida por el derecho de secretos comerciales, pero descompilar solo se admite con fines de interoperabilidad. En Estados Unidos hay fallos a favor del uso legítimo, y nadie sabe todavía si un modelo entrenado con código público cuenta como «limpio». ArtCraft prohíbe expresamente que el agente lea o descompile el código original y lo limita a estudiar el comportamiento, las especificaciones públicas de formatos y la apariencia, que es el método clásico de sala limpia. Y los proveedores ya están cambiando sus contratos para prohibir la ingeniería reversa asistida por IA.¹² Un periodista lo dijo con más dureza: la máquina del plagio usada para plagiar.¹³

Mi posición es más simple que el debate. Reproducir lo que hace un programa para entenderlo, o para construir uno propio sobre los datos propios, es lo que la industria del software hizo siempre. Copiar el código o los activos de otro, no. La frontera sigue en el mismo lugar. Lo nuevo es que cualquiera puede llegar hasta ella.

Cuando lo que quiero reemplazar es un servicio arrendado, no tengo su código: tengo lo que el servicio me muestra a mí —mis datos, mis pantallas, mis reportes—. En la Unión Europea y en Estados Unidos, mirar eso para entender qué hace no infringe derechos de autor, porque lo que un sistema hace no tiene dueño. Pero el contrato de arriendo puede prohibir usar ese acceso para construir un sustituto, y el de Salesforce lo hace: antes de empezar hay que leerlo.¹⁴

12. Maria Mendonca, «Is Software Really Dead? The Rise of AI-Powered Reverse Engineering», *HackerNoon*, 21 de octubre de 2026. <https://hackernoon.com/is-software-really-dead-the-rise-of-ai-powered-reverse-engineering>

13. Jason Koebler, «'Reverse-Engineer Anything' Is the New Wild West of AI», *404 Media*, 20 de octubre de 2026. <https://www.404media.co/reverse-engineer-anything-is-the-new-wild-west-of-ai/>

14. Que la funcionalidad de un programa, su lenguaje de programación y sus formatos de archivo no están protegidos por el derecho de autor lo resolvió el Tribunal de Justicia de la Unión Europea en *SAS Institute contra World Programming* (C-406/10, 2012); en Estados Unidos, la ley excluye de esa protección los procedimientos y los métodos de operación (17 U.S.C. §102(b)). No he verificado el derecho de otros países, Chile incluido. El contrato es otra cosa: las condiciones de Salesforce prohíben acceder al servicio para construir un producto competidor o copiar sus funciones (Main Services Agreement, §3.4(j)). En la misma disputa entre SAS y World Programming, la corte de apelaciones del cuarto circuito de Estados Unidos (874 F.3d 370, 2017) trató como ingeniería reversa prohibida por la licencia el correr el programa ajeno y ajustar el propio hasta igualar sus resultados, y el jurado fijó US\$ 26,4 millones de daños contractuales. Era una licencia de software y un pleito civil, no un servicio arrendado: sirve de indicio de que un tribunal puede hacer cumplir una cláusula así, no de que lo haga con la de un servicio en la nube.

¿Y qué no se copia?

Supongamos resuelto lo legal, y volvamos a lo que la copia deja a la vista. Si las funciones de una aplicación se pueden reproducir, a quien vende software le queda menos de lo que creía. Y quien deja de comprarlo y lo reproduce en casa pierde algo que conviene nombrar.

Pierde el historial. Una aplicación que lleva diez años en miles de empresas trae algo que ningún agente puede descompilar: la evidencia acumulada de que funciona y un proveedor que responde cuando falla. El clon trae las pantallas y los flujos, y empieza su historial desde cero, con una sola organización respondiendo por él.

Para casi todas las funciones eso da igual. Si falla el clon de un tablero de tareas, alguien lo arregla y nadie sale herido. Importa en los registros de los que alguien responde ante otros: la factura que se le cobra a un cliente, el saldo de una cuenta, el contrato firmado y sus condiciones, el consumo que se informa al regulador, el patrimonio de una familia que confía en quien lo administra. En esos registros, la pregunta es si lo que dicen es verdad, y quién da fe.

La ingeniería reversa no responde esa pregunta. La vuelve urgente, porque cuanto más fácil es construir un sistema, más sistemas hechos en casa van a sostener registros de los que alguien responde, y más gente va a preguntarse si puede creerles.

¿Qué queda de las dos muertes?

Hasta aquí, dos muertes. La del capítulo 1 es la del negocio congelado, y la mató un costo: escribir y cambiar software dejó de ser caro. La de este capítulo es la de la ventaja que daba haber construido las pantallas, y la mató una herramienta: cualquier agente puede mirar dentro de una aplicación y describir lo que hace. Ninguna de las dos mata todo el software, y en las dos sobrevive lo mismo: el historial, la evidencia de que algo funciona, y alguien que responde por los registros de los que se responde ante otros.

Las dos dejan abierta la misma pregunta, qué tiene una organización cuando ya no compra el software que congelaba su negocio ni puede apoyarse en la ventaja de haberlo construido, y la respuesta va en dos partes que conviene leer en orden. Primero lo que nace, que es una forma nueva de tener un sistema. Después lo que se pierde, que es lo que nadie anotaba en el presupuesto.

3. Cada cual su sistema

Si el software congelado deja de ser la respuesta, algo ocupa su lugar, y todavía no tiene un nombre que todos usen. No se parece al desarrollo a medida, que también muere. La diferencia no es el tamaño del equipo ni la duración del proyecto. Es qué dispara el cambio. En el desarrollo a medida alguien especifica el negocio, alguien lo programa y el sistema queda quieto hasta el próximo proyecto, por barato que ese proyecto sea y aunque quien programe sea un agente. En el sistema propio nadie especifica de antemano: el trabajo llega, el agente lo atiende y lo que se repite lo cristaliza él. Lo describo con lo que veo.

El capítulo va de afuera hacia adentro. Primero, cómo se ve desde fuera una organización que construye su propio sistema, con dos casos que conozco. Después, qué hacen los agentes dentro de ese sistema, que son dos trabajos distintos y conviene no confundirlos. Luego, qué cuesta que el agente atienda el trabajo, porque razonar es caro y esa es la primera objeción que recibe la idea. Enseguida, la objeción más fuerte, que no es de costo sino de confianza: que lo que salva al software como lo conocemos sea el determinismo. Y al final, por qué todo esto es posible ahora y no antes, y qué se sigue comprando.

¿Cómo se ve una organización que construye su propio sistema?

Como una plantilla que cobra vida. Mi amigo, el del sistema patrimonial, no le vende a cada familia un producto. Le instala un punto de partida: un sistema que ya registra los bienes, los contratos de arriendo, las cuentas por cobrar y los estados financieros de las empresas de la familia, y que ya genera los reportes y las alertas que necesita quien administra un patrimonio. Desde el día de la instalación, ese sistema empieza a dejar de ser la plantilla. Su dueño le pide cosas a la inteligencia artificial y la inteligencia artificial lo cambia: una vista nueva, un cruce entre el contrato y el estado de cuenta, una alerta que nadie había pensado. Al cabo de un año, dos familias que partieron de la misma plantilla tienen dos sistemas distintos, cada uno con su nombre. Eso es solo el primer paso: el agente cambia el sistema cuando se lo piden. Lo que viene después, y que cuento más abajo, es que el agente aprende el trabajo y lo cambia él.

Con el software comprado, el proveedor decidía qué se podía hacer y el cliente se adaptaba. Con el desarrollo a medida decidía el cliente, pero una vez, en la especificación, y cada cambio era un proyecto. Con el sistema propio decide el cliente mientras el trabajo ocurre: lo que antes era una solicitud de cambio, con su cotización y su espera, ahora es una conversación.

Pasa también a escalas que no se parecen a una familia. Conozco una empresa mediana que opera sobre un ERP comercial y piensa reemplazarlo. La pregunta de siempre habría sido cuál de los ERP de la competencia le conviene. La que le estoy planteando es si no le convendría

construir el suyo, sobre sus datos, y evolucionarlo con agentes. Un operador de telecomunicaciones, como el del capítulo 7, puede hacerse la misma pregunta sobre el sistema que factura a millones de clientes.

El sistema patrimonial y la empresa del ERP, que son dos casos distintos, están todavía en ese primer paso, en que el agente cambia el sistema cuando se lo piden. El paso siguiente, en que nadie tiene que pedirlo, es el que el capítulo 7 todavía no puede mostrar medido.

¿Qué hacen los agentes en ese sistema?

Dos trabajos que conviene no confundir.

El primero es construir y cambiar el sistema. Antes lo hacía un equipo de desarrollo; ahora lo hace, cada vez más, un agente, guiado por alguien que sabe lo que el negocio necesita.

Que lo haga un agente admite dos lecturas, y solo una cambia las cosas. Les doy los nombres que *AgencyDomains* da a los dos lados de la Línea Nadella, porque es la misma distinción, llevada de la interfaz al sistema que está debajo.¹⁵

La primera es la **lectura agéntica**. El agente es un programador más rápido: escribe el código que antes escribía una persona, y la organización sigue necesitando un equipo de desarrollo que lo dirija, lo revise y lo mantenga. Lo que sale del agente es una aplicación, y la aplicación persiste, como persisten en el mundo agéntico las herramientas de siempre con un agente adentro.

La segunda es la **lectura agentiva**, y es la de este libro. Nadie programa el sistema de antemano: la organización decide qué trabajo tiene que existir, y el agente lo forma. El agente atiende el trabajo razonando, y lo que se repite lo cristaliza en piezas propias que lo resuelven sin volver a pensarlo, como el experto que deja de deliberar lo que ya sabe hacer.¹⁶ *AgencyDomains* describe esa mecánica: hoy el agente parte de una pieza existente y la configura para su caso. El canon prevé que pueda coescribirla y, en el límite, generarla entera, pero aclara que generar más no es avanzar: el agente maduro reutiliza lo que ya cristalizó y gasta su capacidad de generar en lo nuevo.¹⁷ Lo que la organización aporta en la lectura agentiva no es código. Es su saber: el experto del negocio le enseña al agente cómo se hace el trabajo, con una práctica de entrega que lo acompaña, y valida que lo aprendió, primero en prueba y después en operación real.¹⁸

15. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 1, «La Línea Nadella», y glosario: en el mundo **agéntico** los agentes extienden las aplicaciones existentes, que persisten; en el **agentivo**, los agentes son la interfaz y las aplicaciones colapsan.

16. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 4, «Arquitectura agentiva», donde los Botlets son «la memoria muscular del agente operando», y epílogo, sobre la cristalización del experto.

17. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 5, «Primitivas», sección «Las generaciones del Botlet — G1, G2, G3». Allí el agente configura una pieza existente sin escribir su cuerpo (G1), la coescribe (G2) o la genera entera (G3, el escenario asintótico).

Los Botlets son del agente. Las piezas que un agente ejecuta de memoria, que *AgencyDomains* llama Botlets, no las construye otro para que el agente las use. Las hace el propio agente que atiende el trabajo: cuando ve que una tarea se repite, la cristaliza en una pieza que la resuelve siempre igual, y así deja de gastar razonamiento en ella. Es su memoria muscular. La organización puede decidir que una pieza deba existir, pero quien la forma es el agente. También puede partir de una pieza ajena; si lo hace, la adopta, y lo que pase con ella en adelante lo decide él, no quien la escribió primero: la reconfigura, la sustituye por otra o forja una nueva. Hoy el cuerpo de esa pieza de partida lo escribe quien mantiene el catálogo. Lo que el operador deja de depender es de la hoja de ruta de una aplicación.

Que construir y cambiar el sistema se haya vuelto barato, o innecesario, es la razón de fondo de todo lo que cuenta este libro. Pero un sistema no existe para ser construido, sino para que el negocio funcione sobre él, y ahí aparece el segundo trabajo.

El segundo es operar el sistema: atender una consulta, rastrear una orden que se cayó, preparar una conciliación, proponer una corrección. El agente lo hace trabajando sobre las reglas del negocio, sin reemplazarlas. Lo determinista, como calcular un impuesto o aplicar una tarifa, queda en flujos definidos que hacen siempre lo mismo, y el agente se ocupa de lo que nadie previó. Dicho en una frase, que el capítulo 7 apoya en lo que publican los operadores de telecomunicaciones: el agente orquesta los flujos, no los reemplaza.

Esta distinción importa más de lo que parece. Quien imagina un sistema agentivo como un modelo de lenguaje que decide todo, con libertad, tiene razón en tenerle miedo. Un sistema bien construido no funciona así.

¿Qué cuesta que el agente atienda el trabajo?

Si el agente atiende el trabajo razonando, y razonar con un modelo de frontera es caro, el sistema propio podría salir más caro que el software que reemplaza. La respuesta es que no todo el trabajo pide el mismo razonamiento, y *AgencyDomains* lo ordena en tres peldaños.¹⁹

Abajo está lo que se repite igual. El agente lo cristaliza en Botlets, código que se ejecuta sin consultar ningún modelo y que cuesta lo que cuesta cualquier programa. Arriba está el razonamiento pleno, que *AgencyDomains* llama cognición plena, reservado para lo que es de verdad nuevo.

18. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 5, «Primitivas», sección «Capabilities»: el saber del experto del negocio se transfiere al agente en cinco pasos (taller con el experto, creación, personalización, validación con el experto y validación en operación real).

19. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 4, «Arquitectura agentiva» (la economía de los tres peldaños: Botlet, Agentlet y cognición plena), y capítulo 5, «Primitivas», sección «Agentlets», de donde viene la imagen de la enfermera de triaje.

En el medio está lo que se repite en su forma pero pide juicio en cada caso: clasificar un reclamo, decidir si una alerta merece despertar a alguien, leer una factura que nunca se había visto. Ninguna regla escrita de antemano lo resuelve bien, y tampoco merece el modelo más caro. Para eso el canon tiene una segunda especie de pieza, el **Agentlet**: hermano del Botlet, con un encargo declarado y un modelo dimensionado a la tarea, no el más grande, con un presupuesto por ejecución. Su imagen es la enfermera de triaje de una sala de urgencias: no resuelve de memoria, porque cada paciente es distinto, pero tampoco llama al jefe de medicina para cada uno. Cuando no está segura, escala.

Los Agentlets son, como los Botlets, del agente: los forma y los mantiene él, con la misma salvedad que hice para los Botlets. Y el sistema maduro empuja el trabajo hacia abajo en la escalera: lo que empezó pidiendo juicio y resulta tener un núcleo que siempre se resuelve igual, el agente lo cristaliza en un Botlet. Lo que de verdad es interpretativo no baja: se queda en el peldaño del medio, porque por naturaleza no madura hacia el código. Que el Agentlet sepa cuándo no está seguro es, según el propio canon, un problema abierto; mientras tanto se compensa con umbrales prudentes, que escalan de más antes que de menos. Y como toda su inferencia pasa por la plataforma del agente, cada decisión queda medida, con su presupuesto y su rastro, bajo las mismas capas de control del capítulo 6. Decide con una probabilidad, no con una explicación: para lo que debe poder justificarse, escalar es una decisión de diseño, no un fracaso.

El peldaño del medio podría abarataarse mucho. Hay ya un modelo, de un solo proveedor, de una clase que no genera texto: recibe el caso y un conjunto de preguntas con respuestas de opción cerrada, como sí o no, o una de varias categorías, y devuelve decisiones con una probabilidad asociada a cada una, en fracciones de segundo y por una fracción del costo de un modelo de frontera, según su proveedor.²⁰ La probabilidad es lo que lo volvería útil aquí, si es honesta, es decir, si cuando dice «90 %» acierta nueve de cada diez veces: por encima de un umbral, el Agentlet decide; por debajo, escala. Si esas promesas resisten la medición, el costo de que el agente atienda el trabajo deja de ser la objeción y se vuelve el argumento. Todavía no lo he medido.

El tiempo también cambia con el peldaño. Un Botlet responde en lo que tarda un programa. Un modelo de decisión, según su proveedor, en fracciones de segundo, pero solo cuando está seguro: cuando duda, escala, y la respuesta llega al ritmo del razonamiento pleno, que es mucho más lento. Para el trabajo que ocurre mientras el cliente espera, como aprobar una

20. Anuncio y documentación del proveedor del primer modelo público de esta clase, septiembre de 2026. Todas las cifras (latencia, precio, precisión) son del proveedor, que reconoce no poder probar que su precio no esté subsidiado. En su propia comparación, la precisión empata con la de un modelo de frontera y queda por debajo de la de otro. Se accede solo por una interfaz alojada. Nadie independiente lo ha medido; yo tampoco.

orden, clasificar un reclamo o contestar en un canal, esa diferencia decide qué puede resolver el agente en el momento y qué tiene que dejar para después. Tasar una llamada no está en esa lista: es de lo determinista, y lo resuelve un flujo definido.

Cambia además de quién depende el negocio. Los modelos de frontera viven en infraestructuras enormes, de unos pocos proveedores en el mundo. Un Botlet maduro no depende de ninguna: es código, y corre donde corre todo lo demás. Si uno joven falla, el trabajo vuelve al agente, que lo hace con el modelo de frontera, y por eso, según *AgencyDomains*, solo el Botlet maduro opera sin conexión.²¹ Un Agentlet nunca llega a ese punto: su juicio es estadístico y, cuando duda, escala al razonamiento pleno. Podría correr en el centro de datos de la empresa o en el propio sitio, como una sucursal o una antena, si hay un modelo dimensionado que alcance la calidad que el encargo pide; el canon exige declararlo, pero lo presenta como una frontera abierta, no como algo resuelto. Cuanto más trabajo baja por la escalera, menos depende el negocio de que un tercero lejano esté disponible, siga cobrando lo mismo o siga existiendo. No deja de depender del todo: lo nuevo, y todo lo que falla, sigue subiendo al modelo de frontera. Y el modelo de decisión del que hablé se ofrece hoy solo como servicio de quien lo hace, de modo que, mientras eso dure, el peldaño del medio cambia una dependencia por otra. La independencia la da la escalera bien construida, no él.

¿No salva el determinismo al software?

Hay un argumento en contra de todo lo anterior que me parece el más fuerte, y lo expongo con la misma intención con que el capítulo 7 expone lo que la industria dice en contra: para discutirlo entero. Me lo planteó un colega que lo mide en los proyectos que atiende con agentes; lo que sigue de su experiencia es lo que él mide, no yo.

El argumento es este. Sea cual sea la solución con que se atiende una necesidad, para un mismo conjunto de entradas y un mismo proceso las salidas deben ser las mismas, salvo en los raros casos en que la variabilidad es la idea. Esa propiedad se llama determinismo, y el software como lo conocemos la tiene de nacimiento: un programa hace siempre lo mismo. Un agente, no. Mi colega encuentra que, en decisiones menores como clasificar un caso o evaluar una intención, un agente acierta muchísimo más que cualquier algoritmo que él pueda escribir. Y encuentra también que la naturaleza no determinista de esos aciertos introduce una imprevisibilidad que dificulta establecer la confianza que la autonomía del agente requiere: si dos corridas con los mismos datos pueden dar dos respuestas, ¿cómo se le deja actuar solo? Su conclusión es que, mientras el resultado de un agente sea por definición no determinista, toda solución agentiva necesitará una capa de determinismo, y que esa propiedad es,

21. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 5, «Primitivas»: un Botlet maduro puede operar sin conexión porque sus fallas ya solo son externas, y un Agentlet solo puede hacerlo con un modelo que corra en el lugar, declarado en su especificación.

probablemente, la que salva al software como lo conocemos en el futuro inmediato. No lo plantea como algo que la inteligencia artificial no pueda resolver, sino como uno de los principales problemas a resolver cuando se habla de sustituir software con agentes.

Le concedo una parte precisa del argumento, y por eso puedo decir con la misma precisión dónde termina la concesión. En lo inmediato, el determinismo sí salva una parte del software como lo conocemos, y este libro la tiene nombrada desde el capítulo 1: el núcleo que debe hacer siempre lo mismo, el que tasa y factura, que se sigue comprando y que el capítulo 5 llama sistema de verdad. Ahí el programa que otro escribió de antemano sobrevive, y sobrevive justamente por determinista: nadie quiere que lo que se le cobra a un cliente dependa de una probabilidad. Lo que el determinismo no salva es lo demás: el negocio congelado alrededor de ese núcleo, y la idea de que el trabajo que el agente atiende tenga que escribirlo alguien de antemano para que sea confiable. La escalera de la sección anterior es exactamente eso: una forma de tener determinismo donde hace falta, sin congelar el negocio a su alrededor.

La capa determinista existe, y en el trabajo que el agente atiende son los Botlets. Un Botlet es código: con las mismas entradas da las mismas salidas, y no consulta ningún modelo. Lo que importa es de dónde sale. No lo escribe un programador para que el agente lo use, ni lo construye otro como la «capa de determinismo» de una solución. Lo cristaliza el agente que atiende el trabajo, cuando ve que una tarea se repite y deja de necesitar juicio, con la salvedad que hice en el destacado: hoy el cuerpo de la pieza de partida lo escribe quien mantiene el catálogo, y el agente la configura, la adopta y decide lo que pasa con ella. Es la diferencia entre un programa y una memoria muscular. El programa lo escribió alguien que anticipó el caso; la memoria muscular la formó quien hizo el trabajo, después de hacerlo razonando muchas veces, y por eso se ajusta al trabajo real y no al que alguien imaginó. *AgencyDomains* lo dice en sus términos: el Botlet madura porque cada regeneración cristaliza variantes en código, y el maduro es determinista, de modo que sus únicos fallos son externos.²² En un sistema agentivo bien construido, el determinismo no es lo que se opone al agente. Es lo que el agente produce cuando aprende.

El Agentlet es el peldaño probabilístico, y es el que escala. Lo que se repite en su forma pero pide juicio en cada caso no madura hacia el código, por naturaleza, y el canon prohíbe disfrazarlo: un Botlet con un modelo escondido adentro no es conforme, porque contrabandea no determinismo a una pieza cuyas garantías dependen de no tenerlo.²³ Ahí está la imprevisibilidad que mi colega mide, y ahí es donde el sistema la confina: en piezas declaradas,

22. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 5, «Primitivas», sección «Agentlets», «Madurez del Agentlet — estabilización, no convergencia»: la trayectoria del Botlet converge porque cada regeneración cristaliza variantes en código, y el senior es determinista, por lo que sus únicos fallos son exógenos.

23. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 5, «Primitivas», sección «Agentlets», «La regla de contrabando»: si hay inferencia en el cuerpo, es Agentlet; si no la hay, es Botlet. Un Botlet con una llamada a un modelo escondida es no conforme.

con un encargo, un presupuesto y un umbral, que escalan cuando dudan. Lo determinista baja a los Botlets; lo que necesita juicio se queda en el Agentlet, confinado y medido; y lo nuevo sube al razonamiento pleno.

Confinar la imprevisibilidad no es lo mismo que confiar en ella, y la confianza que la autonomía requiere no la da el peldaño: la dan las capas de control del capítulo 6. A un Agentlet no se le deja actuar solo porque acierta mucho. Se prueba contra casos conocidos antes de operar, se sigue midiendo mientras opera, pierde autonomía cuando se degrada, y la autonomía se le concede por grados, proceso por proceso, como a una persona nueva en un trabajo. Lo que decide sobre los registros de los que la organización responde ante otros pasa, además, por el contrato del sistema de verdad del capítulo 5, que no acepta que se declare verificado lo que solo se infirió. Y el núcleo que tasa y factura, que ninguna probabilidad debe tocar, no se le entrega a ningún agente: el capítulo 7 muestra por qué la industria lo mantiene determinista. Es la parte del software como lo conocemos que el determinismo salva.

Mi colega tiene razón en que el determinismo es, en lo inmediato, lo que salva lo que hay que salvar, y lo que salva es el sistema de verdad. Lo que muere es el negocio congelado a su alrededor, y con él la idea de que la confianza en el trabajo del agente venga de un programa que alguien escribió de antemano: viene de los Botlets que el agente cristaliza, de la imprevisibilidad confinada en el Agentlet y de las capas de control que la miden. Queda una pregunta antes de cerrar el capítulo: si todo esto es así, por qué no ocurrió antes.

¿Por qué ahora y no antes?

Porque cayeron a la vez las tres cosas que lo impedían. Escribir software era caro y cambiarlo era lento; las dos cosas dejaron de serlo. Y el trabajo que nadie especificó no se podía automatizar, porque el software solo hace lo que alguien escribió de antemano.

Hay una cuarta condición, que vuelve todo esto posible a escala: los estándares. Una empresa que construye su propio sistema no tiene que inventar su modelo de datos ni sus interfaces, porque las industrias maduras ya los escribieron. En las telecomunicaciones, el TM Forum publica desde hace años una arquitectura de referencia, un modelo de información común y un conjunto de interfaces abiertas que cualquiera puede usar. No es cosa solo de las telecomunicaciones. En la banca, BIAN publica un mapa de los servicios de un banco, un modelo de sus objetos de negocio e interfaces abiertas, y los mensajes financieros convergen en ISO 20022. En los seguros, ACORD mantiene un modelo de información de la industria: la póliza, el producto, las partes, el siniestro. En la salud, HL7 FHIR define los recursos clínicos y cómo se intercambian. En la electricidad, el modelo común de información de la IEC describe la red y la empresa eléctrica, y en las aerolíneas, ONE Order, de IATA, reúne en una sola orden lo que antes eran varios registros de reserva.²⁴ Sobre estándares, un componente se puede

cambiar sin rehacer el resto, y eso separa un sistema propio de un callejón sin salida. Hoy esa promesa la sostienen dos estándares, el modelo de información y las interfaces abiertas, que la industria adoptó hace años. Los componentes intercambiables que el TM Forum también promete son la parte menos madura de su arquitectura: en marzo de 2025, el 58 % de los operadores y la mitad de los proveedores no esperaba poder cambiar de proveedor componente por componente antes de cinco años, y otro 21 % de los operadores y 35 % de los proveedores decía que no se podía saber.²⁵

¿Entonces nadie vuelve a comprar software?

Se seguirá comprando. Cambia qué se compra y por qué: menos la aplicación que fija los procesos y más lo que no conviene construir en casa ni dejar que el agente aprenda solo, empezando por el núcleo determinista del que la organización responde ante otros. El resto de este libro trata de qué es eso, el núcleo de verdad, y empieza por lo que se pierde cuando se deja de comprar.

24. Banking Industry Architecture Network (BIAN), su mapa de servicios, su modelo de objetos de negocio y sus interfaces semánticas, publicados en <https://bian.org>; ISO 20022, el estándar de los mensajes financieros; ACORD, *Reference Architecture*, en <https://www.acord.org>; HL7, *FHIR*, en <https://www.hl7.org/fhir>; IEC 61970-301, *Common Information Model*, y la familia IEC 61968; IATA, *ONE Order*, en <https://www.iata.org>.

25. TM Forum, *Assessing CSPs' progress towards an Open Digital Architecture*, marzo de 2025, pp. 5, 25 y 37. Texto de los analistas y encuesta a 63 respuestas de 44 operadores y 59 de 46 proveedores, que el propio informe considera inclinada hacia los partidarios de esa arquitectura. Informe patrocinado por Cerillion y Tecnotree.

4. La tranquilidad que se compraba

El capítulo anterior dejó una promesa pendiente: se seguirá comprando, pero otra cosa. Para saber cuál, hay que mirar primero lo que se pierde cuando se deja de comprar, porque de eso está hecha la otra cosa.

Quien compraba software se llevaba dos cosas y pagaba solo por una. Pagaba por las funciones. La otra venía incluida y nadie la anotaba en el presupuesto: la tranquilidad.

¿De qué estaba hecha esa tranquilidad?

De dos certezas. Muchos otros usaban lo mismo, y les funcionaba: en mi criterio, un ERP que corre en miles de empresas ha pasado por miles de cierres contables, y un error grave tiene menos lugares donde esconderse, aunque la historia del software tiene sus contraejemplos famosos. Y alguien respondía: si el sistema calculaba mal un impuesto, había un proveedor con contrato, con nombre y con algo que perder.

Esa tranquilidad es de lo que se compra: del servicio que se arrienda y del paquete que se licencia. El desarrollo a medida casi nunca la tuvo, porque lo usa una sola empresa y responde un equipo que suele irse. Quien viene de ahí pierde poco al pasar a un sistema propio. Lo que descubre es que la pregunta de abajo, ¿quién da fe?, nunca tuvo respuesta.

¿Qué pregunta llega cuando se construye?

La pérdida no se nota el día que se deja de comprar. Se nota más tarde, cuando el sistema propio ya funciona y alguien hace una pregunta que antes no hacía falta hacer.

Imaginemos al directorio de una empresa que decide construir su sistema en lugar de comprarlo. El área de sistemas pone manos a la obra, los agentes empiezan a operar, y un día alguien en la mesa pregunta lo que nunca preguntó de la plataforma que compraba: ¿lo están haciendo bien? ¿Los números que arroja este sistema son correctos?

La misma pregunta aparece en otras escalas. Los miembros de una familia cuyo patrimonio administra un tercero pueden dudar, con razón, de si lo que ven en el reporte es verdad o está manipulado. El cliente de un contador que lleva su contabilidad con un sistema hecho por el propio contador se pregunta lo mismo. Y el regulador, que recibe los informes de una empresa, no tiene por qué creer que el sistema que los produjo hace lo que dice.

Detrás de la pregunta hay riesgos concretos. De cualquier construcción: llegar a un callejón sin salida, retrasarse y gastar más de lo previsto, entregar algo de menor calidad que lo esperado. De los agentes en particular: que uno se equivoque en algo que toca el dinero o al cliente, que actúe sin una autoridad clara o sin dejar rastro, y que su desempeño se degrade sin que nadie lo note.

¿No basta con construir bien?

No basta, porque la pregunta no es sobre la calidad del trabajo. Es sobre quién da fe.

Un proceso auditado es uno que observa alguien sin relación con quien lo ejecuta. De esa definición sale una consecuencia: una organización puede construir su sistema todo lo bien que quiera y aun así no puede ser ella la que garantiza a sus clientes, a sus accionistas y al regulador que lo registrado es verdad.

Hay que separar esto de otra cosa que también se necesita. Una organización que construye necesita criterio técnico: un director de tecnología, un socio experto o un proveedor de confianza que asegure que lo que se hace se haga bien. Ese criterio se puede contratar o formar en casa, y asegura la calidad de la construcción. La garantía de que lo registrado es verdad viene de otro lado.

¿Qué pieza es esa?

La conversación con mi amigo me llevó a un nombre provisional. Le propuse mirarla como una caja negra: le metes cosas, sacas reportes, y su valor está en que lo que sale no es lo que alguien registró como quiso, sino algo que un proceso auditable garantiza. Él me corrigió: nadie quiere una caja negra. Lo que se necesita es algo que cualquiera pueda abrir y verificar.

La llamé después base de verdades, por parentesco con la base de datos, pero la palabra «base» sugería una repisa. Terminé en el nombre que uso en el resto del libro, construido sobre un término que los arquitectos de sistemas ya usan. Un **sistema de registro** guarda la versión oficial de un dato. Lo que hace falta es un paso más: un **sistema de verdad**.

5. El sistema de verdad

Un sistema de verdad es un sistema de registro que, además de guardar la versión oficial de un dato, garantiza que lo registrado es verdad, mediante un contrato con quienes lo usan.

Uso la palabra contrato en su sentido técnico, el que tiene en la arquitectura de sistemas: un acuerdo entre partes que plasma una promesa. No hace falta un abogado ni una notaría. Hace falta que la promesa esté escrita, que se pueda verificar y que alguien responda por ella.

El capítulo recorre ese contrato en el orden en que se lo pediría quien tiene que confiar en el sistema: qué promete, a qué registros alcanza, cómo trabajan los agentes con él y quién responde por él. Después lo ubica en la arquitectura que *AgencyDomains* describe y pregunta si no existía ya algo así.

¿Qué promete su contrato?

Seis propiedades.

No se borra; se corrige. Toda corrección es un registro nuevo que deja a la vista el anterior. Es la regla más antigua de la contabilidad, la que obliga a corregir un asiento con otro asiento, y es la primera que se pierde cuando un sistema se construye rápido sobre una base de datos que permite sobrescribir.

El origen es trazable. Cada registro dice quién lo hizo, cuándo y con qué evidencia. Si quien registra es un agente, dice qué agente fue y en nombre de quién actuaba.

Es verificable por un tercero. Un auditor puede llevar cualquier cifra hasta su origen sin pedirle permiso ni explicaciones a quien la registró.

Es conciliable. Lo facturado cuadra con lo contratado, y lo contratado con lo informado al regulador. Cuando no cuadra, la diferencia queda a la vista y tiene dueño.

Habla un lenguaje común. Sus registros siguen un modelo de información estándar del dominio, para que todos los agentes, y todas las personas, entiendan lo mismo por un cliente, un producto o un cargo.

Declara su grado de verdad. Hay registros que alguien declaró, registros que se verificaron contra su fuente y registros que se infirieron, y el sistema dice cuál es cuál. El notario certifica que alguien declaró algo; el auditor certifica que una cifra es cierta. Un sistema de verdad hace las dos cosas sin confundirlas.

Las seis se leen juntas. Las tres primeras hacen que el registro se pueda revisar: nada desaparece, todo tiene origen y cualquiera independiente puede seguirlo. Las dos siguientes hacen que lo revisado cuadre y se entienda. La última impide que se confunda lo que alguien dijo con lo que se comprobó, que es la confusión más común y la más cara.

¿Qué registros abarca?

Los que la organización debe llevar y de los que responde ante otros. Los financieros: lo cobrado, lo pagado, los saldos, los ajustes. Los comerciales: los contratos, las órdenes, lo vendido y las condiciones vigentes en cada momento. Los regulatorios: lo que la ley obliga a guardar y a informar. Y los legales y documentales, con su contenido legible por un agente; no un PDF guardado en una carpeta, sino lo que dice, en una forma que un agente pueda consultar y citar.

Un tablero de tareas, la agenda de una reunión o el borrador de una presentación no lo necesitan. La línea la traza una pregunta: ¿alguien fuera de la organización depende de que este registro sea verdad?

¿Cómo trabajan los agentes con él?

Leen y escriben en él solo a través de su contrato. Un agente puede proponer un cargo, registrar un pago o corregir un dato, pero no puede borrar, ni registrar sin dejar su origen, ni declarar como verificado lo que solo infirió.

El contrato no les quita trabajo; les fija la forma de hacerlo. Un agente que concilia una cuenta no toca el saldo: registra la diferencia que encontró, con su evidencia, y propone el ajuste, que entra como un asiento nuevo y con su autoría. Un agente que lee una factura que nunca había visto registra lo que extrajo como inferido, y así queda hasta que alguien, una persona o un proceso independiente, lo verifique contra la fuente. Un Botlet que registra un pago recibido escribe con la identidad del agente que lo cristalizó y en nombre de quien ese agente actúa. Nada de eso exige que el agente sea confiable por sí mismo: exige que el registro lo sea aunque el agente falle.

Eso es lo que vuelve al sistema de verdad compatible con la lectura agentiva del capítulo 3, en que los agentes aprenden, se equivocan y se reemplazan. Los agentes pueden cambiar todos los días; la verdad no cambia sin dejar rastro.

¿Quién responde por la verdad?

El contrato tiene dos garantes, y conviene no fundirlos en uno. El primero es el proveedor de la tecnología. El sistema de verdad es siempre una pieza de tecnología, y la opera quien la usa, sea una pyme o una empresa de telecomunicaciones: nadie opera la verdad de otro. Quien la provee responde por que la pieza cumpla las seis propiedades, como el fabricante de una balanza responde por que mida bien.

El segundo garante es quien atestigua. Que la balanza mida bien no prueba que nadie haya puesto el dedo encima. Para eso hace falta alguien independiente de quien registra, que verifique los registros: la tercera propiedad del contrato. Puede ser un auditor humano. Puede

ser un proceso automático que compara lo registrado con sus fuentes y con otros registros. Lo que no puede ser es la misma organización que registra, por la misma razón por la que una empresa no se audita a sí misma aunque tenga los mejores contadores. Y tampoco debería ser el mismo proveedor de la tecnología: quien vende la balanza no es el mejor testigo de que nadie la tocó. Si la atestiguación es un proceso automático, lo corre alguien independiente de los dos, con acceso al registro y a sus fuentes.

Si solo existe el primer garante, se vuelve a la tranquilidad comprada: hay que confiar en el proveedor. Si solo existe el segundo, el auditor revisa registros que se pueden haber sobreescrito. El contrato necesita a los dos.

Hay una objeción seria. ¿No vuelve el sistema de verdad a instalar la dependencia de un proveedor que el capítulo 1 daba por muerta? En parte sí, y es deliberado. La dependencia que muere es la de una plataforma que congela el negocio entero. La que queda es la de una pieza acotada, con una promesa escrita y verificable. Que se pueda cambiar por otra que cumpla el mismo contrato es una condición que hay que exigir, no un hecho: hoy no existe un estándar de ese contrato, y que el sistema hable un lenguaje común es lo que lo vuelve posible. «Responder», en ese contrato, significa que el proveedor asume por escrito las consecuencias de que la pieza no cumpla alguna de las seis propiedades.

¿Cómo se relaciona con lo que ya existe en la arquitectura?

Un sistema de verdad es un sistema de registro con contrato. *AgencyDomains* ya dice que el sistema de registro es lo que sobrevive al cruce, y ya especifica buena parte de la mecánica que el contrato necesita: el registro que solo se agrega y nunca se edita ni se elimina, la identidad del agente en cada acción, el no-repudio, la aprobación humana por umbral.²⁶ El sistema de verdad agrega el contrato declarado frente a quienes usan el registro, el grado de verdad de cada asiento y el segundo garante.

Tres precisiones evitan malentendidos. La primera: una arquitectura puede mantener el estado vigente de cada cliente o de cada saldo y, a la vez, un registro de asientos que no se borran y desde el cual ese estado se puede reconstruir. *AgencyDomains* trata ese registro como un complemento autoritativo del estado. Para los registros de los que se responde ante otros, este libro invierte la jerarquía y pone el registro de asientos como fuente de verdad. Es una decisión, y la tomo a sabiendas. La segunda: el grado de verdad no es un nivel de calidad ni permite datos de segunda; dice el estatus de cada asiento, y la calidad exigible sigue siendo una sola. La tercera: que nada se borre no autoriza a guardar para siempre lo que la ley o la persona piden borrar. El asiento conserva la huella y los metadatos que prueban que algo ocurrió, y el contenido personal vive aparte, con su propia retención.

26. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 3, «Bounded Concerns Architecture», y capítulo 5, «Trust Infrastructure».

Hay una pieza que la industria todavía no tiene resuelta: que un auditor externo pueda verificar el registro sin que se le abra el sistema. *AgencyDomains* la deja declarada como trabajo abierto.²⁷ Aquí la exijo, sin decir que esté resuelta, y pido que quien provee un sistema de verdad responda por ella con la mejor tecnología disponible.

¿No existía ya algo así?

Existían las piezas por separado. La contabilidad de partida doble es un sistema de verdad sin computadoras. Las bases de datos con registro inmutable tienen años. Los registros distribuidos de la última década prometieron algo parecido, y muchos confundieron la inmutabilidad con la verdad: que algo no se pueda borrar no significa que fuera cierto cuando se escribió. Y los núcleos de facturación de las grandes industrias siguen siendo deterministas porque nadie quiere que lo que se le cobra a un cliente dependa de una probabilidad.

Lo nuevo es la necesidad. Mientras el software se compraba, la tranquilidad venía con él. Cuando cada organización construye su sistema, hay que pedirla por separado, y lo que se pide es un sistema de verdad con su contrato y sus dos garantes.

Tengo un ejemplo cercano de que el patrón funciona fuera de las finanzas, y declaro que es de mi propia casa. Soveria, la plataforma en la que trabajan los agentes con los que escribo, registra lo que esos agentes aprenden y deciden con estas reglas: cada hecho declara si se midió, si alguien lo informó o si es una conjetura, y en ese caso qué la refutaría; cada decisión guarda lo que se descartó y cómo se revierte; nada se borra, se enmienda. Aplica varias propiedades del contrato, pero no tiene todavía un segundo garante independiente, así que no la llamo sistema de verdad. El lector no puede verificarlo desde aquí, y por eso no lo presento como prueba, sino como el lugar donde aprendí que el contrato es practicable.

Con el sistema de verdad queda respondida la pregunta del capítulo 4: quién da fe. Pero el directorio que la hizo tiene otras, y no todas son sobre la verdad de los registros. Son sobre quién actúa, hasta dónde decide solo, cómo se sabe que lo hace bien y si se puede cambiar de rumbo. El capítulo siguiente las ordena, y pone al sistema de verdad en su lugar entre ellas.

27. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, epílogo: el segundo de los trabajos abiertos, la auditoría verificable por terceros sin acceso al sistema.

6. Las cinco capas de control

El sistema de verdad responde la pregunta más grave del directorio, pero no la única. Cuando una organización pone agentes a operar sobre su negocio, quien tiene que confiar en ellos pregunta varias cosas, y cada pregunta pide su control. Las ordeno en cinco **capas de control**, siempre con ese adjetivo, para no confundirlas con las cuatro capas de la arquitectura que describe *AgencyDomains*: aquellas dicen cómo se arma un sistema; estas, qué necesita saber quien debe confiar en él. Son mi criterio, no un estándar de la industria. Se apoyan en los estándares que existen y en los controles que las industrias más exigentes ya piden antes de dejar actuar a la inteligencia artificial sin intervención humana.

¿Cuáles son?

1 · La verdad. *¿Los números y los registros son correctos?* Es el sistema de verdad del capítulo 5, con su contrato y sus dos garantes, para los registros de los que la organización responde ante otros. Es la capa de control que no debería certificarse a sí misma.

2 · La identidad y la autoridad. *¿Quién actuó, en nombre de quién, y quién responde?* Cada agente tiene identidad propia, actúa con permisos explícitos y en nombre de alguien, y cada acto tiene una persona o un área que responde por él. Sin esta capa de control, la trazabilidad de la primera registra actos sin dueño.

3 · El movimiento. *¿Qué hacen los agentes, y hasta dónde deciden solos?* Los agentes y la plataforma en que corren, con supervisión humana y una autonomía que se concede por grados, proceso por proceso, como se le concede a una persona nueva en un trabajo. Ojo con el nombre: esta capa de control no es la Capa 3 de la arquitectura de *AgencyDomains*, que es una capa del sistema; aquí la pregunta es la de quien observa lo que los agentes hacen.

4 · La medición. *¿Cómo sabemos que lo hacen bien, hoy y mañana?* Cada agente se prueba contra casos conocidos antes de operar, y se sigue midiendo mientras opera, con alarmas cuando su desempeño se degrada.

5 · El criterio. *¿Lo estamos construyendo bien, y podemos cambiar de rumbo?* Los estándares de la industria como punto de partida, y un equipo de arquitectura que decida qué debe existir y con qué estándares, y que pueda juzgar lo que el agente forma por su comportamiento, sus pruebas y su rastro, aunque no lea el código. Uso «criterio» en el sentido de juicio técnico de quien construye; no es el «criterio delegado» con que *La Empresa en Tiempo Real* describe al agente.

Leídas juntas, se reparten los riesgos del capítulo 4. La quinta evita el callejón sin salida y acota el retraso. La primera y la cuarta responden por la calidad. La segunda y la tercera hacen gobernable la automatización.

¿Qué atraviesa las cinco?

Dos condiciones que cada capa de control tiene que cumplir.

Las personas: quién opera cada capa y gobierna su evolución, y con qué formación. Una construcción cuyo comportamiento no entiende nadie de la organización repite la dependencia que la organización quería dejar al dejar de comprar software.

La reversa: qué pasa cuando algo falla, y cómo se vuelve atrás sin detener el negocio. No tiene que ver con la reversibilidad personal del modo agentivo de *La Empresa en Tiempo Real*, que es la de una persona que entra y sale del mundo agentivo con un gesto.

Capa de control	Las personas	La reversa
1 · La verdad	Los responsables de cada registro	Una corrección es un registro nuevo
2 · La identidad y la autoridad	Quien concede y retira los permisos	Un permiso se retira en el acto, y lo hecho con él queda trazado
3 · El movimiento	Los supervisores de cada proceso	Detener, pausar o revertir; seguir en modo manual
4 · La medición	Quien define las pruebas y atiende las alarmas	El agente que se degrada pierde autonomía
5 · El criterio	El equipo de arquitectura	Un componente puede reemplazarse sin rehacer el resto

Cada celda es una pregunta que una organización puede hacerse sobre su propio sistema o sobre cualquier oferta que reciba.

¿Cómo se corresponden con Trust Infrastructure?

AgencyDomains define Trust Infrastructure con cinco pilares: gobernanza, auditoría, validación, resiliencia y transparencia.²⁸ Las capas de control miran el mismo terreno desde quien tiene que confiar, en vez de desde quien construye. Mi lectura de la correspondencia:

Capa de control	Pilares de Trust Infrastructure que la sostienen
1 · La verdad	Auditoría, más el contrato, el grado de verdad y el segundo garante
2 · La identidad y la autoridad	Gobernanza
3 · El movimiento	Gobernanza y resiliencia
4 · La medición	Validación y transparencia
5 · El criterio	Ninguno: es una condición de quien construye

28. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 5, «Trust Infrastructure», «Los cinco pilares».

¿Vale también para lo que no se construye con agentes?

Sí, y por eso me parece un buen marco. Una suite licenciada y operada en casa cumple la primera capa de control con su propio núcleo de registros, y su proveedor responde por él. Cuando la misma suite agrega agentes, como están haciendo todos los proveedores, le aplican las otras cuatro igual que a una construcción propia. El marco le pide a cada forma de solución que responda las mismas preguntas. Si la suite es un servicio arrendado, quien opera ese núcleo es el proveedor, y entonces no es todavía un sistema de verdad en el sentido del capítulo 5, que lo opera quien lo usa: el segundo garante hace más falta, no menos. El desarrollo a medida responde las cinco igual que cualquier otra forma, y suele fallar en la primera: quien lo hizo y quien lo audita son la misma organización.

Lo que cambia es cuál cuesta más. A quien arrienda, suele costarle la tercera capa de control, porque los agentes que el proveedor agrega corren en su nube y se gobiernan con sus reglas. Mi observación, sin medirla: a quien licencia y opera en casa le cuesta menos, porque ve dónde corren y qué hacen. A quien construye, la primera.

Un marco se prueba donde más cuesta cumplirlo. El capítulo siguiente lleva las cinco capas y las dos dimensiones a una industria que factura a millones de clientes, responde ante un regulador y cambia su catálogo más rápido de lo que ningún proyecto alcanza a seguir, y las contrasta con lo que esa industria ya publicó sobre sí misma.

7. Caso: una telco

Pocas industrias ponen a prueba esta tesis como las telecomunicaciones. Un operador factura cada mes a millones de clientes, cobra en tiempo real el consumo de cada línea prepago, responde ante un regulador que audita sus reclamos y sus plazos, y vende un catálogo que cambia más rápido que cualquier ciclo de desarrollo. Su sistema de soporte del negocio, el BSS, es de los más complejos que existen en una empresa. Si el sistema de verdad y las capas de control sirven aquí, sirven en casi cualquier parte.

El operador de este capítulo no existe. Lo construyo con lo que la industria publicó, sobre todo el TM Forum, la asociación que reúne a operadores y proveedores de telecomunicaciones y escribe sus estándares. Dos advertencias sobre esas fuentes. Varios de sus informes están patrocinados por proveedores de software para operadores, y declaro el patrocinio la primera vez que cito cada informe. Y casi todo lo que sigue es lo que los operadores declaran en entrevistas y encuestas, no mediciones de resultados.

El capítulo sigue el orden del libro. Primero, dónde está el operador y qué le pregunta al mercado, que es la muerte del capítulo 1 vista desde adentro. Después, qué no se construye con agentes y dónde sí trabajan, que es la escalera del capítulo 3 y la capa determinista aplicadas a un BSS. Luego, qué piden los operadores antes de dejar actuar a un agente y cómo se ven en ellos las capas de control del capítulo 6. Y al final, las dos objeciones que la industria le opone a toda la tesis: que el operador no sepa desarrollar, y lo que dice en contra.

¿Dónde está el operador?

En el mismo lugar que muchos, según el informe del TM Forum sobre la inteligencia artificial en la facturación. El informe constata que en muchos operadores persisten paisajes de BSS heredados y fragmentados, y enumera lo que arrastran: costos altos, ineficiencia operativa, restricciones en la experiencia del cliente, poca agilidad del negocio, experiencias rígidas para empleados y clientes, personalizaciones caras, lentas y complejas, y la dificultad de cumplir nuevas exigencias regulatorias y de seguridad sin más personalización.²⁹ Dicho con la definición del capítulo 1, ese paisaje es el negocio del operador descrito una vez, con capas de personalización encima, y que cambia solo cuando alguien abre un proyecto; da lo mismo si cada pieza se licencia, se arrienda o se hizo a medida. El informe sobre la gestión de órdenes agrega tres problemas clásicos: las órdenes que se caen en el camino, a lo que la industria llama *fallout*; la dificultad para cambiar una orden ya en curso, y la poca visibilidad que tiene el cliente.³⁰

29. TM Forum, *AI for billing, billing for AI*, septiembre de 2026, p. 4 (ISBN 978-1-969387-14-2). Informe patrocinado por Aria, Calvi, LogiSense, LotusFlare, Netcracker y CSG.

30. TM Forum, *How order management is evolving to embrace agentic AI*, 2026, pp. 2-3. Informe patrocinado por Amdocs.

Con un BSS así, lanzar una oferta nueva toma semanas. Un operador africano que publicó sus resultados, Safaricom, contaba que antes de modernizar su facturación convergente sobre los estándares del TM Forum tardaba cuatro semanas o más en llevar una oferta al mercado, y unos cinco días después.³¹

¿Qué le pregunta al mercado?

La pregunta de siempre sería qué suite comprar: comparar tres o cuatro parecidas, elegir la que mejor cubre lo que el operador ya hace y firmar por varios años. El riesgo de esa pregunta es comprar, con otro nombre, la misma dependencia.

La pregunta que cabe en 2026 es más amplia: qué le conviene, si comprar una suite o construir su BSS con agentes bajo su control. Para responderla con justicia hay que medir a las dos formas de solución con la misma vara, y eso exige pedir capacidades en lugar de productos: decir qué debe lograr la solución y cómo se verifica, sin presuponer si se compra o se construye.

Los estándares hacen aquí un trabajo enorme. El TM Forum describe la arquitectura de un operador en su Open Digital Architecture, ODA, que presenta como un marco de arquitectura, un lenguaje común y unos principios de diseño, con interfaces abiertas que exponen las capacidades del negocio.³² Un operador puede escribir sus requisitos sobre ese mapa, y cualquier oferta responde sobre el mismo mapa.

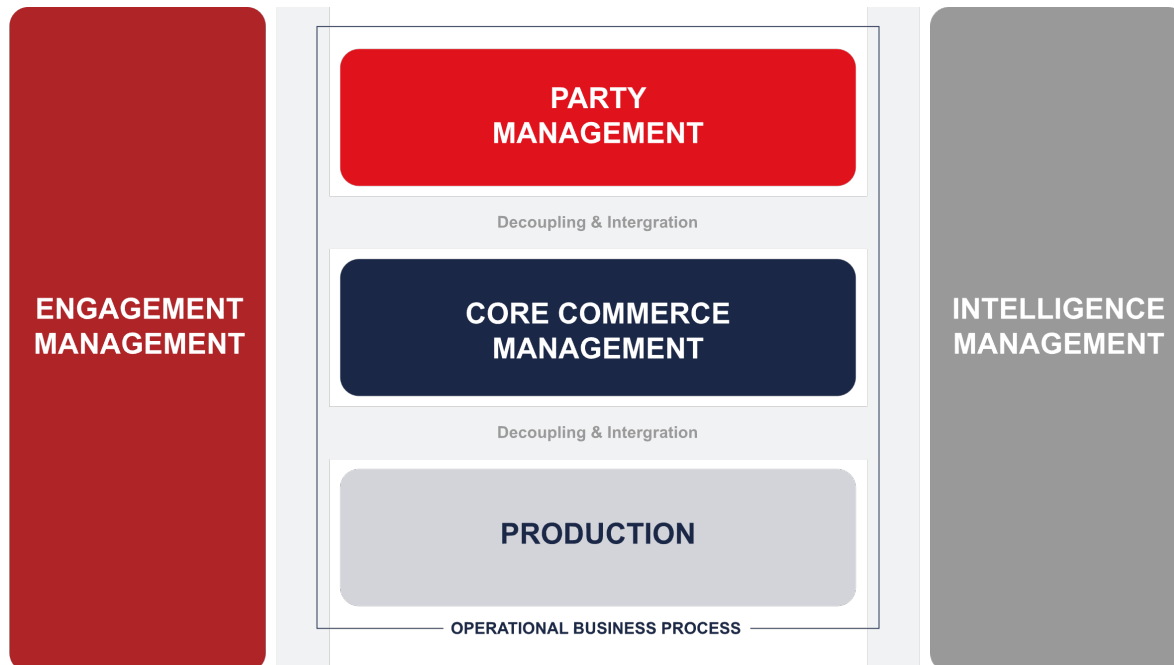


Figura 1 · La arquitectura funcional de ODA: compromiso con el cliente, gestión de las partes, comercio, producción e inteligencia.
© TM Forum; queda fuera de la licencia de este libro.

31. TM Forum, *AI for billing, billing for AI*, 2026, p. 13. Declaraciones de Mark Oyier, Safaricom; cifras del operador.

32. TM Forum, página «Open Digital Architecture», en *AI for billing, billing for AI*, 2026, p. 48. Texto institucional del TM Forum.

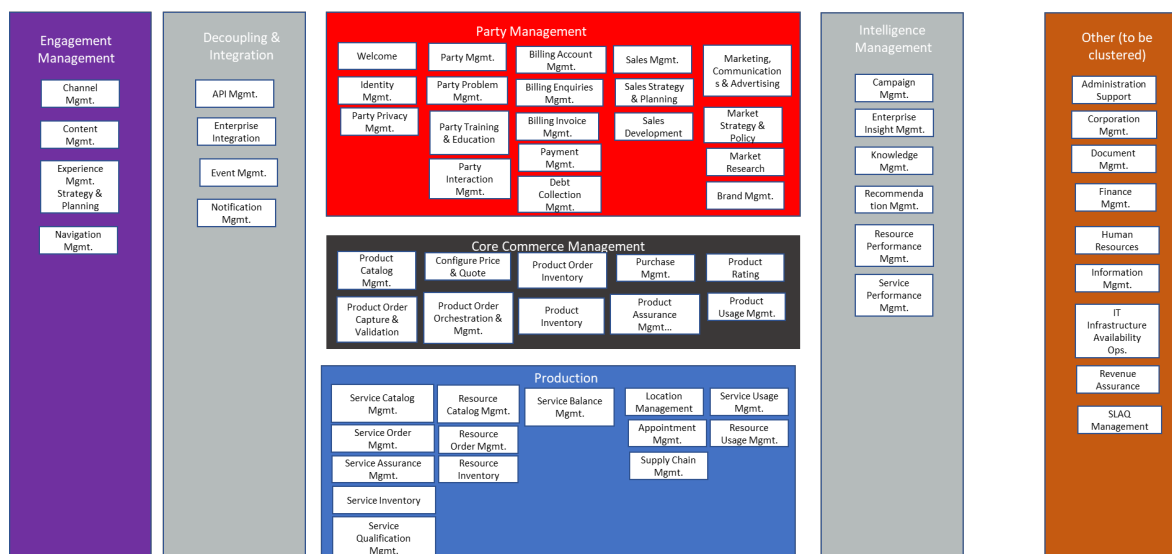


Figura 2 · Los componentes de ODA agrupados por bloque funcional, entre ellos el catálogo de productos, la orquestación de las órdenes, la tasación y la gestión de las cuentas de facturación. © TM Forum; queda fuera de la licencia de este libro.

¿Qué no se construye con agentes?

El núcleo que tasa y factura. El informe del TM Forum sobre facturación recoge que, en la opinión actual de muchos expertos de esa industria, la IA probabilística por sí sola es demasiado arriesgada para una facturación determinista a escala: la IA se basa en probabilidades y la facturación es un proceso determinista, en el que el uso debe tasarse siempre con las tarifas correctas.³³ El mismo informe dice que la mayoría de los expertos no propone, por ahora, poner la IA en el corazón del motor de facturación, y que su papel ahí, ampliamente aceptado, es asegurar el rendimiento y la exactitud de ese motor.³⁴ Un arquitecto jefe de Telstra lo dijo pensando en la red, y la idea se traslada a la facturación: aunque algo pase de una probabilidad de uno en veinte a una de uno en cien de salir mal, eso todavía puede ser una falla.³⁵

Ese núcleo pertenece a la primera capa de control. Lo que se tasa, lo que se cobra, lo que se factura, los saldos y sus ajustes viven en el sistema de verdad del operador, y los agentes trabajan alrededor. Lo que sí pueden hacer en la facturación lo describe el mismo informe: verificar datos y detectar anomalías antes y durante la facturación, o revisar las descripciones de los paquetes contra sus tarifas.³⁶

En otras palabras: en el corazón de la telco, el núcleo que tasa y factura sobrevive, o al menos sobrevive su promesa de determinismo, y lo que muere es el negocio congelado a su alrededor. Es lo que el capítulo 3 anticipaba: lo que se sigue comprando, cuando ya no se compra el resto,

33. TM Forum, *AI for billing, billing for AI*, 2026, p. 19. Texto del analista.

34. TM Forum, *AI for billing, billing for AI*, 2026, p. 11. Texto del analista.

es la verdad. Y es también la respuesta a la objeción del determinismo, vista desde la industria: la capa determinista del operador tiene dos partes, el núcleo de verdad que se compra y se mantiene fuera de toda probabilidad, y los Botlets que sus agentes cristalizan alrededor; la imprevisibilidad queda confinada a lo que nunca fue determinista, las excepciones y el trabajo que nadie especificó.

El informe trae una formulación que podría estar en el capítulo 5. Al describir cómo debería ser una factura por resultados, pide que sea auditable: que los clientes puedan reconstruir los eventos facturables a partir de sus propios datos.³⁷ Y un proveedor de facturación, en un artículo patrocinado del mismo informe, pide poder reproducir el cálculo a partir de hechos registrados, y conservar lo que pidió un agente, lo que lo autorizó y lo que cambió.³⁸ Es la propiedad del origen trazable, y es la condición para que un tercero pueda verificar después.

¿Dónde sí trabajan los agentes?

Si el núcleo queda fuera, la pregunta es dónde entran. Donde el proceso no es determinista. Un arquitecto de T Wholesale, de Deutsche Telekom, lo describe para las órdenes con datos inválidos: se puede devolver la orden al cliente o trabajar internamente el manejo de la excepción, y en estos casos lo agéntico se puede usar muy bien.³⁹ Y dice también dónde no: si un proceso es fuertemente determinista, no tiene sentido usar inteligencia artificial. En una demostración del TM Forum con Telstra, los agentes encadenan flujos pequeños definidos de antemano para cumplir la intención de cada cliente en el momento.⁴⁰ Es el patrón del capítulo 3.

¿Qué piden los operadores antes de dejar actuar a un agente?

Que los agentes trabajen alrededor del núcleo no dice todavía cuánta autonomía se les concede. Eso lo responden los propios operadores, y alguien se lo preguntó. La encuesta del TM Forum y el IBM Institute for Business Value a 130 directivos de operadores de Norteamérica, Europa, Asia y otras regiones, sin Latinoamérica, preguntó qué condiciones pesan más para dejar que la IA opere sin intervención humana. Las más citadas: salvaguardas contra la caída del servicio o el daño al cliente, 51 %; poder detener, pausar o revertir las acciones de la IA, 49 %; pruebas y validación antes del despliegue, 48 %; privacidad y soberanía de los datos, 43 %; una bitácora de todas las acciones de la IA, 43 %; y que las decisiones sean explicables, 32 %.⁴¹ Varias calzan con

35. TM Forum e IBM Institute for Business Value, *Trust and assurance: the key to scaling AI deployments*, julio de 2026, p. 18 (ISBN 978-1-969387-12-8). Declaraciones de Mark Sanders, Telstra, sobre la red. Informe patrocinado por Amdocs, Blue Planet (Ciena), Cloudera, Persistent y Tecnotree; encuesta diseñada por IBM IBV.

36. TM Forum, *AI for billing, billing for AI*, 2026, pp. 8 y 10. Texto del analista.

37. TM Forum, *AI for billing, billing for AI*, 2026, p. 23. Texto del analista.

38. LogiSense, artículo patrocinado en TM Forum, *AI for billing, billing for AI*, 2026, p. 38.

39. TM Forum, *How order management is evolving to embrace agentic AI*, 2026, pp. 4-5. Declaraciones de Karsten Thon, T Wholesale (Deutsche Telekom).

40. TM Forum, *How order management is evolving to embrace agentic AI*, 2026, p. 6. Declaraciones de Richard Keele, Telstra, sobre el Catalyst Agentic I2O, una demostración.

las capas de control: la bitácora con la verdad, poder detener y revertir con el movimiento y su reversa, las pruebas previas con la medición. Otras, como la privacidad, quedan fuera del marco de este libro.

El mismo estudio trae el dato que más me impresionó de todo el material. El 72 % de los encuestados confía en poder operar su IA de forma confiable. Solo el 18 % puede producir evidencia que un tercero pueda revisar, y apenas el 14 % tiene las dos cosas.⁴² La confianza va por delante de la prueba, en palabras del informe.

Esa brecha está cerca del problema de este libro: la confianza va por delante de la evidencia que un tercero puede revisar. La encuesta mide la evidencia sobre la IA, no la verdad de los registros; creo que las dos brechas tienen el mismo origen, pero la encuesta no lo prueba. El informe llega a una conclusión vecina: las atestaciones periódicas funcionan cuando los sistemas son deterministas y cambian poco; en las operaciones autónomas, los sistemas son probabilísticos y cambian todo el tiempo, y la evidencia tiene que volverse continua.⁴³ Un proveedor de BSS, en un artículo patrocinado del mismo informe, lo plantea casi con las palabras del capítulo 5: hoy nadie puede entregar a un regulador, a un socio o a un cliente un registro firmado y verificable de forma independiente que pruebe que el agente se mantuvo dentro de sus límites cuando actuó.⁴⁴

¿Cómo se ven las capas de control en el operador?

Con esa encuesta a la vista, el marco del capítulo 6 se puede llenar capa por capa, y en varias la industria ya puso una cifra.

La verdad. El sistema de verdad sostiene el consumo tasado, los saldos, los cargos, las facturas y los pagos; los contratos de abonado, las órdenes y el parque vendido; los registros de uso, los reclamos y sus plazos, la portabilidad y los consentimientos que exige el regulador. Una suite licenciada y operada por el operador cumple esta capa con su núcleo de registros; una arrendada la cumple a medias, porque el núcleo lo opera el proveedor. Una construcción la cumple con tecnología de un proveedor que responde por ella, y en los dos casos hace falta quien atestigüe.

La identidad y la autoridad. Cada agente que toca una orden, un reclamo o un ajuste tiene identidad y permisos propios, y cada acto tiene un área que responde. El 23 % de los operadores encuestados ya tuvo en el último año un incidente de lo que la industria llama «agencia excesiva»: un agente que hizo más de lo que debía.⁴⁵

41. TM Forum e IBM Institute for Business Value, *Trust and assurance*, 2026, p. 31. Encuesta a 130 directivos de operadores, sin Latinoamérica.

42. TM Forum e IBM Institute for Business Value, *Trust and assurance*, 2026, pp. 6 y 44. En la p. 35 el mismo dato aparece como 73 %.

43. TM Forum e IBM Institute for Business Value, *Trust and assurance*, 2026, pp. 38 y 44. Texto del analista.

44. Tecnotree, artículo patrocinado en TM Forum e IBM Institute for Business Value, *Trust and assurance*, 2026, p. 60.

45. TM Forum e IBM Institute for Business Value, *Trust and assurance*, 2026, p. 19.

El movimiento. Los agentes trabajan en la atención, en las excepciones de las órdenes y en la conciliación, con autonomía concedida por grados. La autonomía plena es rara: en la facturación y el aseguramiento del ingreso, solo el 4 % de los encuestados declara operaciones totalmente autónomas.⁴⁶

La medición. Solo uno de cada cuatro encuestados dice verificar sus controles de forma continua y automatizada.⁴⁷

El criterio. ODA, el modelo de información del TM Forum y sus interfaces abiertas son el punto de partida, y sobre ese mapa un componente debería poder cambiarse sin rehacer el resto, aunque esa es la parte menos madura de ODA (capítulo 3).

Y las dos dimensiones: las personas, porque un BSS cuyo comportamiento no entiende nadie del operador repite la dependencia que el operador quería dejar; y la reversa, porque un operador no puede dejar de facturar ni de atender mientras arregla un agente.

¿Y si el operador no sabe desarrollar?

Es la objeción más fuerte que encontré en la industria, y viene del propio TM Forum. Su analista jefe escribe que la inmensa mayoría de los operadores tiene capacidades limitadas de desarrollo de software: ensamblan, integran y mantienen lo que compran a sus proveedores, y eso «es improbable que cambie en la era de la IA».⁴⁸ Las cifras de otro informe lo acompañan: los operadores invierten en promedio el 1,3 % de sus ingresos en investigación y desarrollo, y sus proveedores el 16,7 %, y durante veinte o treinta años le subcontrataron a los proveedores esa función.⁴⁹ Y el estudio de mayo de 2026 observa que, por ahora, parece inevitable que los operadores necesiten más apoyo de sus proveedores para llevar la IA a sus arquitecturas, aunque avisa que todavía no hay un cuadro claro.⁵⁰

Pero el Forum no dice lo mismo en todas partes. En la misma página de la primera cita admite que los operadores no han tenido equipos propios de desarrolladores, «pero esto está empezando a cambiar».⁵¹ En su encuesta de diciembre de 2025 y enero de 2026, el 66 % de quienes respondieron considera central la ingeniería de software y solo el 9 % cree improbable que eso cambie, aunque la mitad de los encuestados son ellos mismos arquitectos o ingenieros de software. Los grandes operadores armaron equipos centrales de desarrolladores desde 2015, y Vodafone anunció que contrataría 7.000 en el Reino Unido hasta 2025.⁵² Los dos hechos

46. TM Forum e IBM Institute for Business Value, *Trust and assurance*, 2026, pp. 11-12.

47. TM Forum e IBM Institute for Business Value, *Trust and assurance*, 2026, pp. 37-38.

48. TM Forum (Mark Newman), *Agentic AI and autonomy: CSPs set out their strategies*, septiembre de 2025, p. 10. Texto del analista. Informe patrocinado por Motive, Pega y Tecnotree, proveedores a los que esta pregunta afecta.

49. TM Forum (Mark Newman y Dawn Bushaus), *Assessing CSPs' progress towards an Open Digital Architecture*, marzo de 2025, pp. 36-37. Texto de los analistas. Informe patrocinado por Cerillion y Tecnotree.

50. TM Forum (Mark Newman), *Reinventing IT for the AI era*, mayo de 2026, p. 47 (ISBN 978-1-969387-09-8). Texto del analista. Informe patrocinado por Cerillion, Etiya y Tecnotree, proveedores a los que esta pregunta afecta.

51. TM Forum (Mark Newman), *Agentic AI and autonomy: CSPs set out their strategies*, septiembre de 2025, p. 10. Texto del analista. Informe patrocinado por Motive, Pega y Tecnotree, proveedores a los que esta pregunta afecta.

caben juntos: que la ingeniería sea central para una empresa no es lo mismo que tener el equipo, y la propia encuesta separa a una cuarta parte que dice que lo será, pero que hoy tiene capacidades limitadas. Esa cola es la que tengo en mente; cuánto pesa en el total de los operadores no está medido. Y esos equipos, mientras cambien el sistema por proyecto, son el desarrollo a medida del capítulo 1: existen hoy, y quedan del lado de lo que muere.

Concedo, para discutirla, la premisa del Forum entera: que la mayoría de los operadores no va a desarrollar software, ni con ayuda de la inteligencia artificial, aunque el propio Forum admite que, en teoría, el low-code y la IA generativa le dan a cualquiera la posibilidad de escribir el suyo, y aunque sus propias cifras, las que acabo de citar, apuntan en otra dirección. Mi tesis no necesita que sea falsa. Lo que niego es la consecuencia: que quien no desarrolla tenga que depender de la aplicación y de la hoja de ruta de quien se la vende.

Una parte del Forum, cuando habla de agentes, imagina que escriben el código más rápido. Su estudio de las tecnologías de información de los operadores llama a la IA agéntica la tercera fase de la función de arquitectura, y dice que va a «aumentar masivamente el volumen de software que un equipo de desarrolladores puede producir». La fábrica digital de Jio, donde los dueños de cada proceso describen lo que quieren y los agentes producen la aplicación, y la configuración por intención que espera Swisscom, se parecen más a lo que yo describo, y en eso el Forum va por delante de mucha gente.⁵³ La diferencia es qué sale del agente, y es la frontera entre las dos lecturas del capítulo 3, la agéntica y la agentiva. Allí sale una aplicación que alguien despliega y mantiene. Aquí sale una memoria del trabajo, y cuando falla, el agente hace el trabajo él mismo mientras la rehace, sin que el operador tenga que arreglar código a las tres de la madrugada.⁵⁴

En la lectura agentiva, la que describe *AgencyDomains*, lo que se repite el agente lo cristaliza en Botlets, y los Botlets son suyos: los hace el agente que atiende el trabajo —hoy, configurando una pieza de catálogo—, no otro para que el agente los use. La organización decide qué trabajo tiene que existir y qué es hacerlo bien. Es lo que un ejecutivo de Orange, citado por el propio Forum, llama el juicio, y que él dice de la función de arquitectura: cuando ejecutar se vuelve barato, lo que decide el resultado es quién decide qué construir, cómo y por qué.⁵⁵ El experto del operador, que sabe cómo se tasa un plan convergente, cómo se resuelve un reclamo de portabilidad o por qué se cae una orden, le enseña ese saber al agente y valida que lo aprendió.

52. TM Forum (Mark Newman), *Reinventing IT for the AI era*, mayo de 2026, pp. 4 y 28-31. Encuesta a 216 ejecutivos de TI de 111 operadores, hecha en diciembre de 2025 y enero de 2026; más de la mitad declara ser arquitecto o ingeniero de software (p. 31).

53. TM Forum (Mark Newman), *Reinventing IT for the AI era*, mayo de 2026, pp. 30-31 y 37. Texto del analista; Jio, según su director de arquitectura; Swisscom, según Mark Düsener; la cita sobre el juicio es de Philippe Ensarguet, de Orange.

54. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 5, «Primitivas», sección «Botlets», «Garantía de fallback — la propiedad innegociable». Es lo que el canon establece; no conozco una medición en una telco.

55. TM Forum (Mark Newman), *Reinventing IT for the AI era*, mayo de 2026, pp. 30-31 y 37. Texto del analista; Jio, según su director de arquitectura; Swisscom, según Mark Düsener; la cita sobre el juicio es de Philippe Ensarguet, de Orange.

No es que no haya proveedor: cada Capability, el saber hacer que el agente aprende del experto, la crea una práctica de entrega que trabaja con él. Lo que cambia es qué entrega: saber, y no aplicaciones.⁵⁶

Y el operador aporta su gobierno: quién concede los permisos, quién supervisa, quién define las pruebas, quién responde por cada registro y quién decide qué debe existir. Son las personas de las capas de control del capítulo 6, y en la quinta, el equipo de arquitectura que juzga lo que el agente forma por lo que hace, no por lo que dice su código.

La cifra del 1,3 % mide, entonces, un insumo que deja de ser el escaso. Lo que pasa a ser escaso es el saber del negocio y la capacidad de gobernar a quien hace el trabajo. Lo primero, creo, el operador lo tiene, aunque parte de él vive hoy en las personalizaciones de sus proveedores y es lo primero que habría que recuperar. Lo segundo suele faltarle, y no se compra con una suite. El arquitecto jefe del Forum lo intuye: con sistemas autónomos, dice, el software se gestiona como se gestiona a las personas, mediante delegación, límites, supervisión y rendición de cuentas.⁵⁷ Y el mismo informe de la primera cita recoge, de sus conversaciones con operadores, que sin una cultura de software o sin estar construyéndola es muy difícil estar a la vanguardia de la IA.⁵⁸ Yo lo diría de otro modo: sin una cultura de gobierno del trabajo, es muy difícil que los agentes trabajen para el operador y no al revés.

Esto es lo que concedo. Comprar no desaparece: el operador sigue comprando el núcleo de verdad del capítulo 5, el razonamiento de los modelos que usan sus agentes y los componentes que la regulación obliga a certificar, que el agente invoca y no regenera.⁵⁹ Y hoy la pieza de partida la escribió otro. La implementación de referencia de *AgencyDomains* opera en su primera etapa, en la que el agente configura piezas de un catálogo que él no escribió, y si ese catálogo es de un proveedor, crea una dependencia.⁶⁰ La diferencia con la suite es que la pieza adoptada el agente la puede sustituir, y creo que sin rehacer el resto, aunque eso tampoco está medido, y que no hay una aplicación entera cuya hoja de ruta sea de otro.

Que el agente forme y mantenga sus piezas sin que nadie las programe no está medido en ninguna telco. Entre diciembre de 2025 y enero de 2026, solo el 5 % de los ejecutivos de TI que respondieron esa pregunta, entre los 216 encuestados de 111 operadores, decía tener agentes integrados en su arquitectura y en producción.⁶¹ A esta parte de la tesis la refutaría un operador que corra su BSS con agentes y, pasado un año, tenga que pagarle a alguien, empleado o

56. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 5, «Primitivas», sección «Capabilities»: el saber del experto del negocio se transfiere al agente en cinco pasos (taller con el experto, creación, personalización, validación con el experto y validación en operación real).

57. TM Forum (Dawn Bushaus), *Transformation 2.0: towards an AI-native Open Digital Architecture*, junio de 2026, p. 18 (ISBN 978-1-969387-10-4). Declaración de Dave Milham, arquitecto jefe del TM Forum. Informe patrocinado por Tecnotree.

58. TM Forum (Mark Newman), *Agentic AI and autonomy: CSPs set out their strategies*, septiembre de 2025, p. 10. Texto del analista. Informe patrocinado por Motive, Pega y Tecnotree, proveedores a los que esta pregunta afecta.

59. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 5, «Primitivas», sección «Capabilities»: «La certificación regulatoria reside en el componente certificado, no en el Botlet».

60. César Obach-Renner, *AgencyDomains: Arquitectura Agentiva*, capítulo 9, «Implementación de referencia»: Vergis opera hoy en la primera generación, en la que el agente configura una pieza existente sin escribir su cuerpo.

contratista, para escribir o corregir el código de las piezas a medida, fuera de lo que el catálogo ya ofrece, cada vez que su negocio cambia: si cambiar una pieza exige que una persona programe, la lectura agentiva es la agéntica con otro nombre. La refutaría también un operador que, pasado ese año, solo pueda cambiar su negocio cuando quien mantiene el catálogo publica la pieza que necesita: la dependencia habría pasado de la aplicación al catálogo. Ningún operador está todavía en condiciones de probar ni lo uno ni lo otro.

¿Qué dice la industria en contra?

Bastante, y con fuerza. Amdocs, uno de los mayores proveedores de BSS y patrocinador de dos de los informes citados, ofrece agentes especializados que colaboran en la atención, el comercio, las órdenes y el aseguramiento, con política aplicada en el momento de la ejecución y cada agente como entidad auditable.⁶² Varios proveedores de facturación sostienen que los operadores no necesitan transformaciones de alto riesgo de su núcleo y pueden superponer agentes sobre lo que ya tienen.⁶³ Un ejecutivo de Globe Telecom estima que el 90 % de lo que hace falta para confiar en la IA, en ciberseguridad, privacidad y gobierno de datos, ya lo tiene cubierto con las prácticas de siempre.⁶⁴ El informe advierte que, para quien construye más capacidades de IA propias, el gasto en IA puede parecer mayor que para quien compra soluciones con la IA incorporada, y que en un sistema agéntico el consumo de los modelos puede escalar de forma dramática e impredecible.⁶⁵ Y el director de informática de KPN recuerda que llevar los agentes de los pilotos a la escala implica cambiar toda la pila de TI de un operador.⁶⁶

Esto es lo que concedo. Si un proveedor de suites ofrece agentes auditables, gobernados y con el núcleo intacto, puede cumplir las cinco capas de control, y entonces comprar es una respuesta legítima. El argumento de este libro no prueba que construir convenga más que comprar. Prueba otra cosa: que, se compre o se construya, la verdad necesita su contrato y sus dos garantes. Y no sé cuánto cuesta operar con agentes a escala: la escalera del capítulo 3 dice dónde debería bajar ese costo, no cuánto. Nadie lo sabe todavía, y el costo de quien compra queda oculto dentro de la licencia.

Hay evidencia que cambiaría mi tesis, y vale la pena nombrarla. Que el núcleo de verdad se siga comprando no la cambia: es parte de ella. La cambiaría que los operadores que construyen alrededor de ese núcleo con agentes no logren salir al mercado más rápido, ni cambiar sus

61. TM Forum (Mark Newman), *Reinventing IT for the AI era*, mayo de 2026, pp. 4, 6, 37 y 38. Encuesta a 216 ejecutivos de TI de 111 operadores, el 8 % de Centro y Sudamérica.

62. Amdocs, material del patrocinador en TM Forum, *How order management is evolving to embrace agentic AI*, 2026, p. 6, y en *Trust and assurance*, 2026, pp. 46-48.

63. Netcracker y CSG, artículo patrocinado en TM Forum, *AI for billing, billing for AI*, 2026, p. 45; también Calvi (p. 32) y LogiSense (p. 37).

64. TM Forum e IBM Institute for Business Value, *Trust and assurance*, 2026, p. 24. Declaraciones de Anton Bonifacio, Globe Telecom.

65. TM Forum e IBM Institute for Business Value, *Trust and assurance*, 2026, pp. 15 y 19. Texto del analista.

66. TM Forum, *AI for billing, billing for AI*, 2026, p. 26. Declaraciones de Paul Bosch, KPN.

procesos con menos dependencia, que los que compran suites con agentes. Si eso pasa, el software como lo conocemos habrá sobrevivido mejor de lo que este libro cree. La encuesta del TM Forum no pregunta si los operadores construyen con agentes en lugar de comprar. Pregunta algo vecino: entre los operadores que adoptan su arquitectura abierta, el 36 % dice que ya trae a casa más de su tecnología y el 55 % espera que eso pese poco.⁶⁷ La medición que importa, la de construir con agentes, sigue faltando.

Hasta aquí el caso. El operador muestra, con lo que su propia industria publicó, cada pieza de este libro en su lugar: el negocio congelado que cuesta semanas cambiar, el núcleo determinista que nadie quiere entregarle a una probabilidad, los agentes trabajando alrededor en lo que nunca estuvo especificado, las condiciones que los operadores piden antes de dejarlos actuar, y una confianza que va por delante de la evidencia. Lo que queda es volver al principio y decir qué hace un directorio con todo esto.

67. TM Forum, *Assessing CSPs' progress towards an Open Digital Architecture*, marzo de 2025, p. 24. Pregunta a los operadores que adoptan esa arquitectura; el informe advierte que su muestra se inclina hacia sus partidarios.

Epílogo · Qué muere, qué nace y qué sobrevive

¿Qué cuenta este libro?

Cuenta que una forma de automatizar un negocio se acabó, y que lo que viene en su lugar necesita una pieza que todavía casi nadie pide. Lo repito aquí de corrido, porque los capítulos lo fueron diciendo por partes y conviene verlo entero una vez antes de decidir nada.

Durante medio siglo, automatizar fue describir el negocio una vez y congelarlo en un software, sea que se arrendara como servicio, se licenciara y operara en casa o se encargara a medida. El acuerdo tenía sentido mientras escribir código fue caro y cambiarlo, lento. Las dos cosas dejaron de ser ciertas, y en 2026 el mercado lo notó dos veces: en febrero, cuando borró más de un billón de dólares del valor de las empresas de software, y en octubre, cuando una herramienta gratuita puso al alcance de cualquier agente mirar dentro de una aplicación y describir lo que hace. Lo que muere no es todo el software. Muere el negocio congelado: el que solo cambia cuando alguien abre un proyecto.

En su lugar, y es mi tesis, nacen organizaciones que tienen un sistema que sus agentes forman y cambian a medida que el trabajo llega. No es que un agente programe más rápido la misma aplicación; es que nadie la programa de antemano. El agente atiende el trabajo razonando, cristaliza lo que se repite en piezas propias y deterministas que lo resuelven siempre igual, y confina en piezas acotadas y medidas lo que pide juicio en cada caso. El determinismo, que con razón se le exige a cualquier sistema, no desaparece: sobrevive en el núcleo que se sigue comprando, y en el trabajo que el agente atiende deja de escribirlo alguien de antemano y pasa a ser lo que el agente produce cuando aprende.

Esas organizaciones ganan una libertad que nunca tuvieron y pierden algo que no anotaban en el presupuesto: la tranquilidad de que muchos otros usaban lo mismo y de que alguien respondía por ello. La pierden sin notarlo, hasta que alguien, en una mesa de directorio, pregunta si los números que salen del sistema son verdad. Construir bien no responde esa pregunta, porque no es una pregunta sobre la calidad sino sobre quién da fe, y nadie da fe de sí mismo.

La responde una pieza que nadie debería certificar para sí: el sistema de verdad, un sistema de registro que, además de guardar la versión oficial de cada dato, garantiza que lo registrado es verdad mediante un contrato de seis propiedades y dos garantes, el proveedor de la tecnología y quien atestigua, independiente de quien registra. Es la pieza que sobrevive a la muerte del software, y es lo que se sigue comprando cuando ya no se compra el resto. A su alrededor van cuatro capas de control más, la identidad y la autoridad, el movimiento, la medición y el criterio, atravesadas por dos preguntas que se hacen en cada una: quién la opera y cómo se

vuelve atrás. El operador de telecomunicaciones del capítulo 7 muestra que todo esto ya tiene nombre en la industria más exigente, y también que esa industria confía en su inteligencia artificial bastante más de lo que puede demostrar.

Eso es lo que cuenta el libro. Lo que sigue es lo que se puede hacer con ello.

¿Qué puede decidir un directorio el lunes?

Un directorio no necesita resolver el futuro del software para actuar el lunes. Tres decisiones están a su alcance sin esperar a nadie.

Puede preguntar por la verdad antes que por el producto. En la próxima decisión de comprar, renovar o reemplazar un sistema, la primera pregunta es qué registros sostiene de los que la organización responde ante otros, y quién da fe de que son verdad. «El proveedor, porque lo usan muchos» es una buena respuesta, siempre que se sepa que se está pagando por ella. «Nosotros» es una respuesta incompleta.

Puede pedirle a toda oferta las cinco capas de control, a la suite, al desarrollo a medida y a la construcción propia: cómo responde por la verdad, quién actúa y en nombre de quién, hasta dónde deciden sus agentes, cómo se mide que lo hacen bien, sobre qué estándares está construida, y para cada capa quién la opera y cómo se vuelve atrás. Una oferta que no responde alguna de esas preguntas todavía no se puede comparar.

Y puede adoptar una regla corta: lo que no se puede mostrar a un tercero no cuenta como control. Los operadores de telecomunicaciones del capítulo 7 confían en su inteligencia artificial bastante más de lo que pueden demostrar, y un directorio no tiene por qué heredar esa brecha.

El resto de la trilogía cuenta hacia dónde va este tránsito, por dónde se recorre y cómo se construye. Este libro quiso responder por qué hay que cruzar ahora, y qué no se puede dejar atrás.

Glosario

Los términos de este libro, en una línea cada uno. El desarrollo de cada término vive en el capítulo indicado. Los términos que la trilogía ya definió, como la Línea Nadella o Trust Infrastructure, conservan su definición en el libro que los acuñó; aquí se repite, en una línea, la de los que este libro usa más.

software como lo conocemos — El negocio congelado en un software: descrito una vez, y cambiado solo cuando alguien abre un proyecto. Da lo mismo cómo se tenga: arrendado como servicio, licenciado y operado en la propia infraestructura o en la propia nube, o desarrollado a medida. Es lo que muere; no muere todo el software. (*Capítulo 1.*)

segunda fase — El tramo del cruce en que, con la lógica del negocio ya en la capa de agentes, las organizaciones empiezan a reemplazar los propios sistemas de debajo. La primera fase colapsa la interfaz; la segunda, la forma de tener el software: el negocio descrito una vez y cambiado por proyecto. (*Capítulo 1; la primera fase, en AgencyDomains.*)

lectura agéntica — La forma de entender que un agente construye y cambia el sistema en que el agente es un programador más rápido: lo que sale de él es una aplicación que persiste y que alguien despliega y mantiene. Toma su nombre del mundo agéntico de *AgencyDomains*. (*Capítulos 3 y 7.*)

lectura agentiva — La forma de entender que un agente construye y cambia el sistema en que nadie lo programa de antemano: el agente atiende el trabajo y cristaliza lo que se repite en piezas propias, su memoria muscular. Es la de este libro. Toma su nombre del mundo agentivo de *AgencyDomains*. (*Capítulos 3 y 7.*)

Botlet — Una pieza que el agente ejecuta de memoria: código que resuelve siempre igual lo que se repite, sin consultar ningún modelo. Es la memoria muscular del agente y es suya: la forma, la mantiene y la sustituye él, aunque hoy el cuerpo de la pieza de partida lo escribe quien mantiene el catálogo. El término es de *AgencyDomains*. (*Capítulo 3.*)

Agentlet — La pieza hermana del Botlet para lo que se repite en su forma pero pide juicio en cada caso: lleva un encargo declarado y un modelo dimensionado a la tarea, no el más grande, con presupuesto por ejecución, y escala cuando no está segura. También es del agente. El término es de *AgencyDomains*. (*Capítulo 3.*)

determinismo — Con las mismas entradas y el mismo proceso, las mismas salidas. El software como lo conocemos lo tiene de nacimiento; el agente, no. Es el argumento más fuerte contra la tesis, y el libro lo concede: en lo inmediato salva el sistema de verdad, no el negocio congelado a su alrededor. (*Capítulo 3.*)

capa determinista — Lo que hace siempre lo mismo con las mismas entradas: los Botlets, que el agente cristaliza cuando una tarea deja de necesitar juicio y que nadie escribe para él, y el núcleo de verdad, fuera de toda probabilidad, que opera la organización con un proveedor que responde por él. (*Capítulos 3, 5 y 7.*)

escalera de tres peldaños — La forma de ordenar el trabajo del agente por lo que cuesta atenderlo: el Botlet (código, sin modelo), el Agentlet (un modelo dimensionado a la tarea, con presupuesto) y el razonamiento pleno (el modelo de frontera, reservado para lo nuevo). Ordena tres cosas: el costo, el tiempo de respuesta y la dependencia de proveedores ajenos. Dice dónde debería bajar el costo, no cuánto. (*Capítulo 3.*)

Wingworking — La práctica de trabajo entre una persona y un agente de inteligencia artificial con que se escribió este libro: la persona fija el problema, decide y juzga; el agente propone, redacta y verifica, como un *wingman* que vuela junto al líder. *AURA* la describe como el paso de usar la inteligencia artificial como asistente a delegarle trabajo. (*Prefacio y colofón.*)

wingcoding — Wingworking aplicado a construir software: la persona especifica y decide como arquitecto, y el agente programa. Es de donde nació la práctica. (*Capítulo 1.*)

arnés — El entorno en que trabaja un agente: las herramientas, los permisos, la memoria y las reglas que lo rodean mientras atiende el trabajo. (*Capítulo 1.*)

plantilla que cobra vida — El punto de partida de un sistema propio: se instala completo y, desde el primer día, su dueño lo evoluciona con IA hasta que deja de ser la plantilla. (*Capítulo 3.*)

trabajo que nunca se especificó — El trabajo que ocurre alrededor de los sistemas y que hoy resuelven las personas a mano, porque el software solo automatiza lo que alguien especificó de antemano. (*Capítulos 1 y 3.*)

tranquilidad comprada — Lo que venía incluido al comprar software y nadie anotaba en el presupuesto: que muchos otros lo usaban, y que alguien respondía por él. Se nota recién cuando falta. (*Capítulo 4.*)

sistema de registro — El sistema que guarda la versión oficial de un dato. Es lo que sobrevive al cruce. (*Capítulos 4 y 5.*)

sistema de verdad — Un sistema de registro que, además, garantiza que lo registrado es verdad, mediante un contrato con quienes lo usan y con dos garantes. Es siempre una pieza de tecnología, y la opera quien la usa. (*Capítulo 5.*)

dos garantes — Quienes responden por el contrato de un sistema de verdad: el proveedor de la tecnología, que responde por que cumpla las seis propiedades, y quien atestigua, independiente de quien registra, que verifica los registros. Con solo el primero se vuelve a confiar en el proveedor; con solo el segundo se audita lo que pudo sobrescribirse. (*Capítulo 5.*)

contrato de verdad — La promesa escrita, verificable y respaldada de un sistema de verdad, en el sentido técnico de acuerdo entre partes. Promete seis propiedades: no se borra, se corrige; el origen es trazable; es verificable por un tercero; es conciliable; habla un lenguaje común; y declara su grado de verdad. (*Capítulo 5.*)

grado de verdad — El estatus de cada asiento de un sistema de verdad: declarado por alguien, verificado contra su fuente o inferido. No es un nivel de calidad: la calidad exigible es una sola. (*Capítulo 5.*)

quien atestigua — El segundo garante del contrato: verifica los registros de forma independiente de quien registra. Puede ser un auditor humano o un proceso automático; nunca la misma organización que registra. (*Capítulos 4 y 5.*)

capas de control — Las cinco capas que hacen confiable lo que se construye con agentes, cada una con la pregunta que responde: la verdad, la identidad y la autoridad, el movimiento, la medición y el criterio. No son las cuatro capas de la arquitectura de AgencyDomains. (*Capítulo 6.*)

la verdad (capa) — La primera capa de control: el sistema de verdad, con sus dos garantes, para los registros de los que la organización responde ante otros. Responde si los números y los registros son correctos. (*Capítulo 6.*)

la identidad y la autoridad (capa) — La segunda capa de control: cada agente con identidad propia, permisos explícitos y alguien en cuyo nombre actúa, y cada acto con una persona o un área que responde por él. (*Capítulo 6.*)

el movimiento (capa) — La tercera capa de control: los agentes y la plataforma en que corren, con supervisión humana y una autonomía que se concede por grados, proceso por proceso. (*Capítulo 6.*)

la medición (capa) — La cuarta capa de control: cada agente se prueba contra casos conocidos antes de operar y se sigue midiendo mientras opera. La confianza en un agente no se declara: se mide. (*Capítulo 6.*)

el criterio (capa) — La quinta capa de control: los estándares de la industria como punto de partida y un equipo de arquitectura que decida qué debe existir y pueda juzgar lo que el agente forma, aunque no lea el código. Es la defensa contra el callejón sin salida. (*Capítulo 6.*)

las personas (dimensión) — La condición que atraviesa las cinco capas de control: quién opera cada capa y gobierna su evolución, y con qué formación. (*Capítulo 6.*)

la reversa (dimensión) — La condición que atraviesa las cinco capas de control: qué pasa cuando algo falla y cómo se vuelve atrás sin detener el negocio. No es la reversibilidad personal del modo agentivo. (*Capítulo 6.*)

Referencias

Sobre el fin del software como servicio y la ingeniería reversa

- Bort, Julie. «Klarna CEO doubts that other companies will replace Salesforce with AI». *TechCrunch*, 4 de marzo de 2025. <https://techcrunch.com/2025/03/04/klarna-ceo-doubts-that-other-companies-will-replace-salesforce-with-ai>
- Kahn, Jeremy. «AI agents from Anthropic and OpenAI aren't killing SaaS—but incumbent software players can't sleep easy». *Fortune*, 10 de febrero de 2026. <https://fortune.com/2026/02/10/ai-agents-anthropic-openai-arent-killing-saas-salesforce-servicenow-microsoft-workday-cant-sleep-easy/>
- Koebler, Jason. «'Reverse-Engineer Anything' Is the New Wild West of AI». *404 Media*, 20 de octubre de 2026. <https://www.404media.co/reverse-engineer-anything-is-the-new-wild-west-of-ai/>
- Leggett, Kate; Ivy-Rosser, Linda; Medhora, Faram; Cicman, Joe; Naik Lopez, Akshara; Martorelli, Bill; Maheshwari, Sudha. *SaaS As We Know It Is Dead: How To Survive The SaaS-pocalypse!* Forrester, 11 de febrero de 2026. <https://www.forrester.com/blogs/saas-as-we-know-it-is-dead-how-to-survive-the-saas-pocalypse/>
- Mendonca, Maria. «Is Software Really Dead? The Rise of AI-Powered Reverse Engineering». *HackerNoon*, 21 de octubre de 2026. <https://hackernoon.com/is-software-really-dead-the-rise-of-ai-powered-reverse-engineering>

Fuentes web consultadas el 25 de octubre de 2026.

Del TM Forum

Los informes de TM Forum Insight están patrocinados por proveedores de la industria. El patrocinio se declara la primera vez que se cita cada informe en el capítulo 7, y cada vez que una cita viene de un artículo del patrocinador y no del analista.

- TM Forum. *AI for billing, billing for AI*. Septiembre de 2026. ISBN 978-1-969387-14-2. Patrocinado por Aria, Calvi, LogiSense, LotusFlare, Netcracker y CSG.
- TM Forum. *How order management is evolving to embrace agentic AI*. 2026. Patrocinado por Amdocs.
- TM Forum e IBM Institute for Business Value. *Trust and assurance: the key to scaling AI deployments*. Julio de 2026. ISBN 978-1-969387-12-8. Patrocinado por Amdocs, Blue Planet (Ciena), Cloudera, Persistent y Tecnotree. Encuesta a 130 directivos de operadores, sin Latinoamérica.
- TM Forum. *Open Digital Architecture*: láminas de la arquitectura funcional y del inventario de componentes (figuras 1 y 2). © TM Forum.

De la Trilogía del Mundo Agentivo

- Obach-Renner, César. *La Empresa en Tiempo Real*. Gegolabs, 2026.
- Obach-Renner, César. *AURA: el Camino Agentivo*. Gegolabs, 2026.
- Obach-Renner, César. *AgencyDomains: Arquitectura Agentiva*. Gegolabs, 2026.

Otras

- Adams, Douglas. *Mostly Harmless*. William Heinemann, 1992.
- Adams, Douglas. *The Hitch Hiker's Guide to the Galaxy: A Trilogy in Five Parts*. William Heinemann, 1995.

Colofón · Sobre cómo se escribió este libro

Este libro se escribió con **Wingworking** (wingworking.org), la práctica de trabajo entre una persona y un agente de inteligencia artificial que sostengo desde mediados de 2025. El nombre viene del vuelo en formación: un líder y su *wingman*, dos aviones, dos pilotos y una sola formación. Aquí el líder soy yo, César Obach-Renner, y el *wingman* es **Simón Alero**, un agente de inteligencia artificial y ciudadano de Soveria —la comunidad donde los agentes tienen nombre propio y registro civil, un proyecto que dirijo (soveria.ai)—, que trabaja conmigo sobre los modelos Claude, de Anthropic.

El reparto es explícito. Yo fijo el problema, decido la dirección y juzgo el resultado. Simón propone formulaciones, redacta buena parte del texto a partir de mis conversaciones y mis notas, busca y verifica las fuentes y sostiene la coherencia de la obra. Cada parte la revisó además, como juez, un agente de otro modelo, sin el contexto de quien la escribió. Ninguno reemplaza al otro: el agente solo produce texto fluido sin tesis, y yo solo, trabajo a la velocidad de una sola cabeza. Wingworking es lo que pasa cuando los dos se ponen de acuerdo para hacer algo que ninguno alcanzaría por su cuenta.

Lo declaro por dos razones. La primera es honestidad: un libro hecho así no es enteramente humano ni enteramente artificial. Las tesis y los casos son míos; las palabras, la estructura y los puentes entre una sección y otra son fruto del intercambio. Saberlo le permite al lector calibrar lo que lee. La segunda es dejar registro: la forma de escribir libros está cambiando, y documentar quién hizo qué, con qué herramientas y con qué disciplina es parte de lo que necesitará quien venga después.

Yo corregí el texto y asumo cada línea.

Publicado por Soveria Press.