Sistemas multiagente declarativos: de usar IA a construir IA

Introducción: de "usar IA" a "construir IA" — un salto crítico

A finales de 2025, un proyecto de agente de IA de código abierto llamado OpenClaw irrumpió en la escena, atrayendo a millones de visitantes y superando las 100.000 estrellas en GitHub en una sola semana. "Tu asistente, tu máquina, tus reglas": el eslogan encendió un enorme entusiasmo en las comunidades tecnológicas del Norte Global. Casi simultáneamente, varios gigantes tecnológicos se apresuraron a lanzar sus propios productos de "agentes de IA", cada uno afirmando revolucionar la naturaleza misma del trabajo del conocimiento. Para los profesionales de la tecnología en el Norte Global, esto puede representar un ciclo más de emocionante iteración técnica; pero para investigadores, dirigentes de movimientos e intelectuales del Sur Global, una pregunta más de fondo exige atención: **mientras todos celebran los avances técnicos de los agentes de IA, muy pocos preguntan quién posee y controla estas herramientas, y a qué agenda sirven.**

Esta no es una pregunta retórica. Cuando una investigadora del Sur Global usa la interfaz de chat de ChatGPT para asistir su escritura, es una **consumidora** de IA: consume un producto diseñado por otros, trabaja bajo reglas fijadas por otros, en plataformas propiedad de otros, con datos seleccionados por otros. En principio, puede pegar su metodología en el *prompt* y guardar el resultado como archivo, pero se trata de soluciones improvisadas, no de capacidades sistémicas. La lógica de comportamiento del sistema permanece opaca para ella, la incorporación de la metodología es ad hoc y frágil, y la acumulación de conocimiento depende enteramente del esfuerzo manual fuera de la plataforma. Cada conversación es una isla; nada se acumula.

Pero si esta investigadora cuenta con un sistema de investigación compuesto por docenas o incluso cientos de agentes —cada uno con responsabilidades y estándares de calidad claramente definidos, capaces de buscar automáticamente en cientos de páginas web, procesar cientos de documentos, analizar según el marco teórico que ella elija y generar informes de investigación con miles de citas—, entonces ya no es una consumidora, sino una **constructora**. El sistema le pertenece, la metodología es incorporada por ella, el conocimiento acumulado pertenece a su institución, y todo el proceso de investigación es transparente y está bajo su control.

El salto de consumidora a constructora es un paso crítico para que el Sur Global reconquiste la soberanía tecnológica.

El fundamento técnico del que depende este salto se denomina **sistema multiagente declarativo**. Este artículo aborda tres preguntas de profundidad creciente: ¿qué es exactamente la llamada "IA agéntica"? ¿Cómo se **construyen** este tipo de sistemas? Y ¿cómo se los combina con el propio trabajo para determinar **qué** construir? En el artículo anterior de esta serie, propusimos la arquitectura técnica AI4SS (inteligencia artificial para las ciencias sociales) —"un núcleo, tres repositorios, cuatro dominios"—. El sistema multiagente declarativo y su lenguaje de patrones POMASA, tema de este artículo,

constituyen precisamente ese "núcleo": **el entorno de ejecución inteligente**, el motor que impulsa todo el sistema AI4SS. El propósito del sistema es analítico: busca, verifica, razona y sintetiza. Que sus resultados adopten la forma de texto es consecuencia de su método, no su objetivo. POMASA no es en sí mismo un sistema multiagente: es una guía de construcción para edificar uno —un conjunto de patrones arquitectónicos verificados que indican qué construir, mientras que el entorno de ejecución inteligente determina cómo.

Esto no es especulación teórica. El proyecto Global South Insights (GSI) del Instituto Tricontinental de Investigación Social ya ha construido y opera decenas de sistemas multiagente declarativos, el más grande y complejo de los cuales —el Sistema de Producción de Investigaciones— comprende varios cientos de agentes declarativos. Lo que sigue se basa en nuestra propia experiencia.

1. Una nueva forma de software: el sistema multiagente declarativo

Cuando se habla de "IA agéntica", el término suele estar envuelto en una neblina de exageración comercial. Una vez disipada esa neblina, lo que emerge es una realidad de ingeniería directa: un sistema multiagente (MAS, por sus siglas en inglés) es, en esencia, una forma de **software** —no magia, ni el atisbo embrionario de una inteligencia general, sino una nueva forma de software con un marco conceptual, unos principios arquitectónicos y una metodología de ingeniería claros. Comprender este punto es el punto de partida para devolver la IA de la exageración a la realidad material.

1.1 De prompts largos a sistemas multiagente: por qué el MAS es necesario

Consideremos a una investigadora que intenta usar IA para una tarea de investigación compleja. Podría escribir un *prompt* extenso, embutiendo definiciones de roles, métodos de investigación, marcos teóricos, formatos de salida y estándares de calidad en un solo bloque de texto. A medida que la tarea se vuelve más compleja, este *prompt* crece hasta las 30.000 o incluso 50.000 palabras, y entonces surgen tres problemas estructurales imposibles de evitar.

Primero, la **dilución de la atención**. Incluso sin exceder la ventana de contexto del modelo, los modelos de lenguaje de gran escala (LLM) enfrentados a *prompts* extensos y cargados de detalles simplificarán su ejecución de manera autónoma, ignorando detalles que consideren irrelevantes, lo cual vuelve impredecible la calidad del resultado. Segundo, la **ausencia de aislamiento modular**. Aunque la autora divida mentalmente el *prompt* en "paso uno, paso dos", todo el contenido se carga en un único contexto, y modificar una sección puede afectar inesperadamente a otra —de forma similar a escribir todo el código en una sola función enorme: aunque se divida por comentarios en secciones, sigue siendo un monolito—. Tercero, **el mantenimiento se vuelve intratable**. Sin un paso de parámetros flexible ni un control dinámico del proceso, modificar una guía de estilo exige recargar el *prompt* completo de 30.000 palabras.

Estos no son problemas hipotéticos. Cualquier investigadora que haya intentado codificar un marco analítico completo —definiciones de roles, jerarquías de fuentes, compromisos epistemológicos, estándares de citas, guías de estilo— en una sola conversación de IA se ha topado exactamente con esto: la IA empieza a ignorar los estándares de citas; modificar el marco analítico desestabiliza el formato de salida; cada nueva conversación requiere recargar todo desde cero. La especificación de conocimiento cuidadosamente diseñada ha desbordado el contenedor de la conversación única.

Para quienes tienen experiencia en ingeniería de software, este dilema resulta familiar. Un prompt largo es, en esencia, concatenación de texto —definiciones de roles, metodologías, guías de estilo y otros componentes cosidos en un solo bloque enorme y entregados a la IA en su totalidad—. Es como leerle a alguien en voz alta, de principio a fin, un manual de operaciones completo y esperar luego que ejecute cada tarea de memoria. La ingeniería de software atravesó esta misma etapa en sus primeras décadas —todo el código escrito en un solo archivo— y, a partir de la revolución de la programación estructurada de finales de la década de 1960, pasó décadas evolucionando hacia la arquitectura modular actual: cada módulo

cumple una función distinta, puede iniciarse y detenerse de forma independiente, y colabora a través de interfaces claramente definidas. La ingeniería de prompts actual sigue estancada en la etapa de "todo en una sola olla", y lo que buscan los sistemas multiagente es precisamente completar el recorrido que la ingeniería de software ya ha realizado: dar a cada módulo un ciclo de vida independiente y límites claramente definidos.

La solución: en lugar de confiar todo a una sola conversación con la IA, descomponer una tarea grande en múltiples **agentes** —cada agente es una unidad de trabajo de IA independiente, dedicada a una tarea específica (como "recopilar literatura sobre un tema dado" o "escribir un análisis según un marco especificado")—. Varios agentes, cada uno responsable de su propio rol y trabajando de manera coordinada, constituyen un sistema multiagente.

1.2 Conceptos centrales del MAS

Comprender cómo funcionan los sistemas multiagente requiere asimilar cinco conceptos centrales.

El **plano de agente** (*blueprint*, en la terminología original de POMASA) es un documento de definición en tiempo de diseño. Es un archivo de texto escrito en lenguaje natural —el formato más común es Markdown (una sintaxis de formato ligera ampliamente usada para redactar documentos), aunque incluso un archivo de texto plano sirve— que describe lo que un agente debe hacer: definición de rol, especificación de entradas, descripción de la tarea, especificación de salidas y estándares de calidad. Un plano de agente es para un agente lo que un plano arquitectónico es para un edificio: el plano no es el edificio en sí, sino una descripción completa de este.

La **instancia de agente** es una entidad de ejecución en tiempo de ejecución. Cuando se "lanza" un plano, el sistema crea una instancia con su propia ventana de contexto independiente; esta comprende la tarea, ejecuta operaciones y produce resultados dentro de su propio espacio. Un solo plano puede lanzar varias instancias simultáneamente —tal como un mismo conjunto de planos puede usarse para construir muchos edificios de estructura idéntica.

El **entorno de ejecución** (runtime environment) es la plataforma de IA que ejecuta a los agentes. Sin embargo, a diferencia de los entornos de ejecución de programas tradicionales, un entorno de ejecución de IA (como Claude Code) no es un ejecutor pasivo de instrucciones, sino un **entorno de ejecución inteligente**: puede comprender intenciones expresadas en lenguaje natural, seleccionar de forma autónoma estrategias de ejecución y emitir juicios y ajustes cuando surgen problemas. Un entorno de ejecución tradicional ejecuta mecánicamente secuencias de instrucciones; un entorno de ejecución inteligente comprende la intención y elige inteligentemente el método de ejecución.

El **bus de datos** es el canal a través del cual los agentes se pasan datos entre sí; en la implementación más común actualmente, esto es simplemente el sistema de archivos. El Agente A escribe su salida en un directorio designado; el Agente B lee de ese directorio como entrada. Este diseño aparentemente de "baja tecnología" encarna una consideración profunda: todos los productos intermedios son archivos que los humanos pueden leer directamente —completos, transparentes y trazables—.

Los **datos de referencia** son conocimiento de dominio externalizado: documentos metodológicos, marcos teóricos, estándares disciplinarios, ontologías de dominio, y así sucesivamente. Por ejemplo, GSI mantiene un documento de referencia sobre los fundamentos de la teoría marxista que se comparte entre múltiples sistemas como línea de base analítica común (un futuro artículo de esta serie examinará este documento de referencia en detalle). Compartidos por todos los agentes, pero invariantes entre tareas, los datos de referencia constituyen el fundamento de conocimiento de un MAS y el vehículo de los activos de conocimiento institucional.

En la práctica, cada uno de estos cinco conceptos se corresponde con actividades de investigación concretas. Los planos de agente son archivos Markdown que describen tareas como "recopilar y verificar datos fiscales sobre el sector extractivo de un país" o "analizar impactos distributivos usando un marco epistemológico declarado". Las instancias de agente son las

sesiones de trabajo individuales de IA lanzadas a partir de esos planos: una instancia recopila datos sobre contratos de gas natural, otra recopila simultáneamente datos sobre regalías mineras. El entorno de ejecución es la plataforma de IA inteligente que ejecuta estos agentes. El bus de datos es un conjunto de carpetas —una que contiene datos verificados, otra que contiene análisis temáticos, una tercera que contiene borradores de secciones—, cada una legible e inspeccionable en cualquier momento. Y los datos de referencia comprenden los activos de conocimiento institucional reunidos por el equipo de investigación: documentos de marco epistemológico, guías metodológicas, estándares de estilo y las fuentes primarias que fundamentan el trabajo de cada agente.

1.3 Declarativo frente a imperativo: por qué el enfoque declarativo es más avanzado

Existen dos enfoques fundamentalmente distintos para construir sistemas multiagente.

El enfoque **imperativo** fue el método dominante en los marcos anteriores a 2025 (como LangChain y AutoGen): el comportamiento del agente se define mediante código Python, código que controla *cómo* actúa el agente. Este enfoque requiere programadores, impone una barrera de entrada alta, exige una cantidad considerable de código y hace compleja la depuración.

El enfoque **declarativo** es un paradigma nuevo: el comportamiento del agente se define mediante documentos en lenguaje natural que describen *qué propiedades debe poseer el resultado*, delegando el *cómo* al entorno de ejecución inteligente para que lo determine de manera autónoma en tiempo de ejecución. Tras el lanzamiento por parte de Anthropic de Claude Code en 2025, el método declarativo se convirtió rápidamente en el enfoque predominante, porque Claude Code, como entorno de ejecución inteligente, admite de forma nativa el lanzamiento y la gestión de subagentes mediante planos en lenguaje natural, lo que hace que los marcos imperativos dejen de ser un prerrequisito para construir sistemas multiagente.

Esta distinción no es una cuestión de preferencia estilística, sino de elección arquitectónica, y su esencia radica en **postergar las decisiones del tiempo de diseño al tiempo de ejecución**. Un sistema imperativo determina, en el momento en que el desarrollador escribe el código, cómo manejar una situación dada; un sistema declarativo describe únicamente cómo debería verse el resultado, delegando las estrategias de ejecución concretas al entorno de ejecución inteligente para que las determine de manera autónoma al enfrentar las condiciones reales. Esto no es pereza, sino el reconocimiento de un hecho material: en el momento del diseño, la desarrolladora posee mucha menos información de la que tendrá el sistema de IA en tiempo de ejecución. Tal como un arquitecto no debería anotar en los planos "si mañana llueve, usar el plan B", sino delegar los ajustes tácticos al equipo de construcción inteligente en el terreno.

La diferencia práctica es marcada. Bajo el enfoque imperativo, una investigadora necesitaría a un programador que escribiera código Python especificando, paso a paso, cómo buscar datos, cómo interpretar los resultados, cómo manejar un enlace roto o un PDF que falla al descargarse. Bajo el enfoque declarativo, una investigadora escribe un plano en lenguaje llano que dice: "Recopilar datos fiscales sobre el sector extractivo entre 2015 y 2025, priorizando documentos presupuestarios gubernamentales, informes de conciliación y consultas del Artículo IV del FMI; verificar cada cifra contra al menos dos fuentes independientes; señalar cualquier discrepancia superior al 5%". Describe cómo debería lucir el resultado; el entorno de ejecución inteligente determina cómo lograrlo. La barrera entre la investigadora y un sistema funcional ya no es el código: es la claridad de su propia especificación.

La evidencia cuantitativa proveniente de la práctica respalda esta elección. La práctica a gran escala de PayPal en 2025 encontró que el enfoque declarativo redujo el tiempo de desarrollo en un 60%, triplicó la velocidad de despliegue y disminuyó el tamaño de los planos de agente en un orden de magnitud. Los ingenieros de PayPal concluyeron: "La mayoría de los flujos de trabajo agénticos se componen de patrones comunes —serialización de datos, filtrado, generación

aumentada por recuperación, orquestación de API— que pueden expresarse mediante un lenguaje declarativo unificado en lugar de requerir código imperativo".

La historia ofrece el mejor marco para comprender esta transformación. Del lenguaje ensamblador a los lenguajes de alto nivel, de la programación procedimental a las consultas declarativas de SQL, del despliegue manual a la "infraestructura como código": cada elevación en el nivel de abstracción ha expandido dramáticamente la frontera de *quién puede participar*. El sistema multiagente declarativo es el hito más reciente en esta línea evolutiva: reduce el umbral para "construir sistemas de IA" de "saber escribir código" a "saber describir un problema".

Esto significa: **quien construye ya no necesita ser programador.**

Lo que se necesita para construir es **conocimiento de dominio**: la capacidad de comprender problemas y de articular con claridad "lo que necesito". Para las investigadoras y dirigentes de movimientos del Sur Global, esta transformación conlleva una implicación decisiva: ya no es necesario depender de ingenieros de software para construir las propias herramientas de IA.

2. Cualquiera puede construir: el lenguaje de patrones como guía de construcción

El sistema multiagente declarativo reduce el umbral de *quién puede construir*, pero *cómo construir bien* aún requiere buenas prácticas. Alguien que construya sin experiencia, enfrentando una página en blanco, sigue sin saber por dónde empezar, cuántos agentes definir, cómo deben colaborar entre sí o cómo asegurar la calidad. El lenguaje de patrones existe precisamente para resolver este problema.

2.1 De "saber programar" a "saber describir": una nueva forma de empoderamiento

La liberación central del sistema multiagente declarativo radica en esto: lo que se necesita para construir no es escribir código, sino escribir documentos. Un plano de agente es un archivo Markdown que contiene definiciones de roles, especificaciones de entrada, descripciones de tareas, especificaciones de salida y estándares de calidad. Modificar el comportamiento de un agente significa modificar un pasaje de texto: sin compilación, sin despliegue, sin depuración.

Esto significa que **las expertas de dominio pueden participar directamente en la definición y modificación del sistema**. Una economista política puede leer un plano de agente, comprender qué hace, y modificar su marco analítico —en el lenguaje natural que domina, en lugar de en Python, que no conoce—. La metodología de investigación puede traducirse directamente en comportamiento del sistema.

Esto tiene una importancia particular para el Sur Global. Bajo el modelo tradicional de desarrollo de IA, las investigadoras deben "traducir" sus requerimientos para ingenieros de software, quienes luego los implementan en código. En ese proceso de traducción, la intención inevitablemente se distorsiona y se simplifica. El sistema declarativo elimina este paso de traducción: quien investiga construye el sistema.

2.2 Lenguaje de patrones: mejores prácticas ejecutables

Pero la libertad no produce excelencia automáticamente. Lxs nuevxs constructorxs aún necesitan saber "cómo se ve un buen sistema".

El **lenguaje de patrones** es la respuesta clásica a esta pregunta. Tiene su origen en la arquitectura —Christopher Alexander propuso en su obra seminal de la década de 1970, *A Pattern Language* [Un lenguaje de patrones], un conjunto de patrones

de diseño verificados por la práctica, cada uno resolviendo un problema de diseño recurrente, formando los patrones una red orgánica de relaciones que en conjunto guían a lxs constructorxs en la creación de obras de alta calidad.

Consideremos uno de los ejemplos originales de Alexander: el patrón n.° 159, "Luz en dos lados de cada habitación". Alexander observó que, cuando las personas pueden elegir, siempre se inclinan hacia habitaciones con ventanas en dos lados, mientras que las habitaciones con una sola ventana quedan desatendidas —porque la iluminación unilateral genera contrastes y deslumbramientos agudos que instintivamente producen incomodidad—. La orientación del patrón es entonces: **al distribuir cada habitación, asegurar que tenga muros exteriores en al menos dos lados donde puedan colocarse ventanas, permitiendo que la luz natural entre desde más de una dirección.** Esto es un "patrón": identifica un problema recurrente (la iluminación unilateral produce incomodidad), ofrece un principio de solución verificado (luz en dos lados) y no prescribe métodos de construcción específicos. La ingeniería de software importó con éxito este pensamiento en la década de 1990: los patrones de diseño de la Banda de los Cuatro siguen siendo lectura obligatoria para todo programador.

POMASA es un lenguaje de patrones que hereda esta tradición de Alexander. Consideremos uno de los ejemplos de POMASA: el patrón STR-02, "Bus de datos del sistema de archivos". Cuando múltiples agentes colaboran, una pregunta central es: ¿cómo se transmiten información entre sí? Podría usarse una base de datos, una cola de mensajes o diversas soluciones técnicas sofisticadas, pero entonces los resultados intermedios se vuelven invisibles para los humanos, encerrados dentro de los mecanismos internos del sistema. Este patrón ofrece el principio: **usar el mecanismo más simple posible: archivos y carpetas.** El Agente A escribe su salida como un archivo en un directorio designado; el Agente B lee de ese directorio como entrada. Todos los productos intermedios son documentos que un humano puede abrir y leer directamente. Si algo sale mal en cualquier etapa, abrir la carpeta correspondiente revela exactamente lo que produjo cada agente: no hay caja negra.

En la era de la IA, el lenguaje de patrones sufre una transformación crítica. Los lenguajes de patrones tradicionales estaban escritos para que los humanos los leyeran: los humanos comprendían el patrón y luego lo traducían ellos mismos a código. Pero en un entorno de ejecución inteligente, el propio lenguaje de patrones está escrito en lenguaje natural, y el entorno de ejecución de IA puede comprender el lenguaje natural; por lo tanto, **el propio lenguaje de patrones se vuelve ejecutable**. Se le puede entregar un conjunto de patrones a un sistema de IA, y este puede generar directamente a partir de ellos un sistema multiagente funcional.

Esto redefine lo que significa "código abierto". El código abierto tradicional significaba publicar código para que otros lo ejecutaran.

En la era de la IA, publicar un lenguaje de patrones equivale a publicar código fuente, porque otros pueden entregar los patrones a un sistema de IA y reconstruir una arquitectura equivalente.

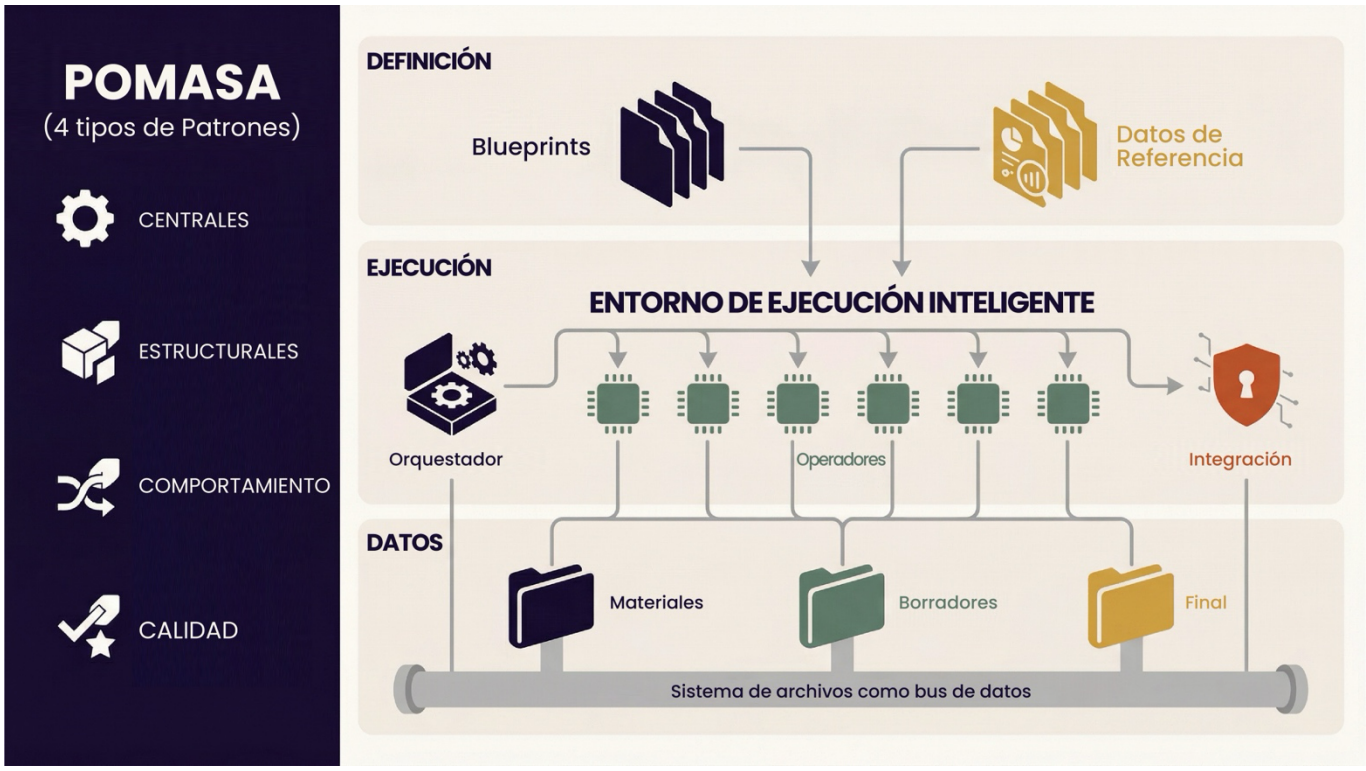
2.3 POMASA: nuestro lenguaje de patrones

POMASA (Pattern-Oriented Multi-Agent System Architecture, arquitectura de sistemas multiagente orientada a patrones) es precisamente ese lenguaje de patrones. No es un sistema de software específico, sino un conjunto de patrones arquitectónicos refinados a partir de la práctica y verificados mediante su despliegue.

POMASA comprende 20 patrones en cuatro categorías:

- **Patrones centrales** (COR, 2 patrones): Agente Definido por *Prompt* (COR-01) y Entorno de Ejecución Inteligente (COR-02): la base de todo lo demás.
- **Patrones estructurales** (STR, 9 patrones): Configuración de Datos de Referencia, Bus de Datos del Sistema de Archivos, Aislamiento del Espacio de Trabajo, Diseño de Agentes Orientado al Negocio, Ensamblaje Componible de Documentos, Guía Metodológica, y otros: definen el esqueleto del sistema.

- **Patrones de comportamiento** (BHV, 6 patrones): Flujo de Trabajo Orquestado de Agentes, Instanciación Fiel del Plano, Ejecución Paralela de Instancias, Refinamiento Progresivo de Datos, y otros: definen cómo opera el sistema.
- **Patrones de calidad** (QUA, 3 patrones): Estándares de Calidad Incorporados, Aseguramiento de Calidad por Capas, Linaje de Datos Verificable: definen cómo el sistema garantiza la fiabilidad.



Panorama de la arquitectura POMASA: desde la definición del plano hasta la ejecución en el entorno de ejecución inteligente, conectados a través del bus de datos del sistema de archivos

Cada patrón se clasifica por necesidad en tres niveles: **Obligatorio**, **Recomendado** y **Opcional**. La configuración mínima viable requiere solo seis patrones obligatorios para empezar; diez patrones recomendados cubren la mayoría de los escenarios; patrones opcionales adicionales pueden introducirse progresivamente según las necesidades reales.

Quien es nuevo construyendo no necesita asimilar los 20 patrones antes de empezar. Los seis patrones obligatorios bastan para transformar una única conversación sobrecargada en un sistema modular: definir agentes en Markdown (COR-01), ejecutarlos en un entorno de ejecución inteligente (COR-02), pasar datos entre ellos a través de carpetas (STR-02), aislar el espacio de trabajo de cada agente (STR-03), incorporar estándares de calidad directamente en cada plano (QUA-01) y orquestar a los agentes en un flujo de trabajo (BHV-01). A medida que crece la confianza, pueden introducirse progresivamente patrones recomendados: ejecución paralela para investigar múltiples dimensiones simultáneamente, aseguramiento de calidad por capas para separar el agente que escribe del agente que revisa.

Este lenguaje de patrones no se diseñó de manera abstracta. Se refinó a partir de sistemas de producción reales, comenzando con 5 patrones y creciendo gradualmente hasta 20 a través de la construcción repetida de sistemas y la resolución de fallos. Cada patrón corresponde a un error cometido alguna vez o a un desafío de diseño enfrentado de manera repetida.

2.4 Cómo empezar en tres pasos

POMASA no es un ejercicio de sillón. Si desea probarlo ahora, solo se requieren tres pasos.

El prerrequisito es un entorno de ejecución inteligente que admita la invocación de subagentes —como Claude Code, Cursor o Cline—. Después de instalar el entorno de ejecución de su elección, ejecute "npx skills add eXtremeProgramming-cn/pomasa" en la línea de comandos para instalar la skill de POMASA (este comando es compatible con todos los entornos de ejecución mencionados). Luego, describa en lenguaje natural lo que desea investigar —por ejemplo: "Ayúdame a crear un sistema de investigación multiagente para analizar las políticas de economía digital en el Sudeste Asiático"—. La IA la guiará en el refinamiento de sus requerimientos, recomendará una combinación adecuada de patrones según su descripción y luego generará automáticamente el conjunto completo de archivos del sistema.

No se requiere una sola línea de código en todo el proceso. El sistema generado es un conjunto de archivos Markdown: planos de agente que se pueden leer, comprender y modificar. POMASA en sí es de código abierto bajo la licencia Apache-2.0. Además, esto no es una pieza de demostración: es el mismo lenguaje de patrones que GSI usa en su trabajo cotidiano.

2.5 Experiencia de GSI: el Sistema de Producción de Investigaciones en la práctica

El **Sistema de Producción de Investigaciones** (RPS, por sus siglas en inglés) de GSI es la validación más exhaustiva de POMASA en la práctica. RPS comprende varios cientos de agentes declarativos y cientos de puertas de calidad, la abrumadora mayoría de ellas bloqueantes: 100% declarativo, sin una sola línea de código de orquestación en Python. Cada agente es un archivo Markdown.

Uno de los productos insignia de RPS es el Informe de Coyuntura de País: un estudio sistemático de la situación político-económica de distintos países. Tomando como ejemplo el informe sobre un país africano, este informe pasó por un flujo de trabajo de múltiples etapas —desde la adquisición e ingesta de literatura, el análisis de coyuntura, la planificación de la investigación, la investigación profunda (con múltiples temas por lote y varios agentes trabajando en paralelo), la redacción integral, la revisión de calidad, la preparación para la publicación, hasta la generación de resúmenes—. Todo el proceso manejó más de 100 documentos de entrada a lo largo de decenas de temas de investigación y produjo un informe integral que contenía miles de citas. El sistema se completó en cuestión de días. Completar un informe de coyuntura de país de calidad equivalente mediante métodos tradicionales requeriría de seis a doce meses. La ganancia de eficiencia asciende a entre 100 y 200 veces.

Pero la eficiencia no es la única medida. El aseguramiento de calidad es la pregunta central que todo MAS declarativo debe responder, porque los sistemas de IA "alucinan" (la investigación ha demostrado que más de la mitad de las citas académicas generadas por ChatGPT son fabricadas, e incluso GPT-4 sigue exhibiendo una tasa de fabricación del 18%). RPS aborda esto mediante cientos de puertas de calidad y decenas de pares productor-validador: el agente que genera contenido no puede revisar su propia salida; la validación debe realizarla de manera independiente un agente separado, comenzando desde cero. El sistema también incorpora un proceso de verificación de hechos en múltiples capas con precisión a nivel de carácter: en un informe de investigación completado, cientos de notas al final lograron cero citas fabricadas. En una evaluación externa, el sistema obtuvo una puntuación alta, con la dimensión de aseguramiento de calidad recibiendo puntuación perfecta y el sistema evaluado como "un sistema listo para producción con innovación significativa".

Esto no es teoría. Es un sistema de producción ya en funcionamiento, que genera resultados de investigación reales.

3. Un ejemplo concreto: de una frase a un informe de investigación

Las dos secciones anteriores expusieron los principios del MAS declarativo y el sistema de patrones de POMASA, pero lxs lectorxs bien podrían preguntar: ¿cómo se ve todo esto en la práctica? Recorramos el proceso completo, desde un único párrafo de requerimientos hasta un informe de investigación terminado que contiene 85 citas.

Un lector que había estado siguiendo la competencia tecnológica entre Estados Unidos y China se topó con siete artículos sobre el tema en internet, que abarcaban la guerra de los chips, la disputa por tierras raras, la energía renovable y la política arancelaria. Cada artículo abordaba una faceta distinta del asunto, pero él quería una visión de conjunto que reuniera estos análisis dispersos, ayudándolo a comprender sistemáticamente el complejo panorama. Su requerimiento central, introducido en la plantilla de entrada de usuario de POMASA, ocupaba un solo párrafo: "La guerra tecnológica de chips entre EE. UU. y China: abarcando chips, IA, robótica, energía, tierras raras y cuestiones relacionadas con la cadena de suministro. Reunir todo esto y complementarlo con un análisis completo de la cadena de suministro, incluyendo la fortaleza manufacturera de China y su ascendencia en la cadena de suministro en áreas adyacentes como vehículos eléctricos, baterías, paneles solares y herramientas de precisión". Proporcionó los siete artículos al sistema como material de referencia.

Introdujo este requerimiento en el generador de POMASA. El generador lo guió a través de varios parámetros clave: preferencias de fuentes de datos (prioridad a documentos de políticas públicas, investigación revisada por pares e informes sectoriales confiables, con atención a fuentes no anglófonas, particularmente chinas); formato de salida (panorama de investigación, de 5.000 a 10.000 palabras); y nivel de aseguramiento de calidad (el nivel más estricto disponible). El generador luego seleccionó automáticamente 13 de los 20 patrones disponibles y produjo un sistema de investigación multiagente completo —siete planos de agente, un conjunto completo de datos de referencia, documentos metodológicos y una estructura de directorios—, todo compuesto por archivos de texto legibles y modificables.

El sistema generado opera como un flujo de trabajo de ocho etapas. El arquitecto narrativo lee los siete artículos de referencia y diseña la estructura narrativa junto con un plan de investigación detallado. Se asignan *investigadores especializados* por tema y se ejecutan en paralelo: para cada tema un agente independiente recopila materiales de internet. El agente verificador de datos revisa cada dato recopilado de manera individual: no una muestra, sino verificación al 100%, un requisito obligatorio de los patrones de calidad de POMASA. Los agentes analistas destilan argumentos a partir de los materiales verificados, organizados por tema. Los agentes redactores de sección redactan cada sección de acuerdo con la guía de estilo designada. Finalmente, el agente auditor de calidad, operando como un rol independiente, realiza una revisión integral desde cero.

Desde completar la plantilla de entrada de usuario hasta generar el sistema multiagente completo tomó aproximadamente media hora. El sistema luego se ejecutó —investigación profunda, verificación de datos, análisis, redacción y auditoría de calidad— durante varias horas, investigando en el proceso cientos de fuentes verificadas de internet. El resultado final fue un informe titulado "Rompiendo el cerco: el ascenso tecnológico de China y el desmoronamiento de la hegemonía tecnológica estadounidense". Con más de 11.000 palabras y 85 citas en línea con URL, cubría las sanciones a los semiconductores y los avances autóctonos, la competencia en chips de IA, las contramedidas sobre tierras raras, el dominio en energía renovable, la manufactura automatizada de "fábricas oscuras" y las implicaciones para el Sur Global. La auditoría de calidad obtuvo una puntuación alta, con la argumentación y la perspectiva del Sur Global recibiendo ambas puntuación perfecta. El auditor simultáneamente produjo una lista concreta de revisiones —identificando áreas de mejora y citas que requerían verificación cruzada adicional—, al tiempo que evaluaba el análisis central como "listo para publicación".

Lo que importa es el patrón, no el caso particular. El mismo proceso se aplica ya sea que el dominio sea la competencia tecnológica, las industrias extractivas en África Oriental, la reforma agraria en Asia Meridional o las condiciones laborales en América Central. Los requerimientos cambian; el conocimiento de dominio cambia; la arquitectura del sistema permanece igual: entran requerimientos, sale un sistema, se produce investigación, con cada paso trazable.

Este no fue un resultado impecable —el propósito mismo de una auditoría de calidad es identificar deficiencias—. Los juicios finales, los refinamientos y las revisiones estructurales aún requieren la participación humana; este es el significado práctico del Humano en el Bucle. Lo que merece atención, sin embargo, es el proceso mismo: el párrafo único de requerimientos de un lector y siete artículos de referencia, combinados con el sistema de patrones de POMASA y sin

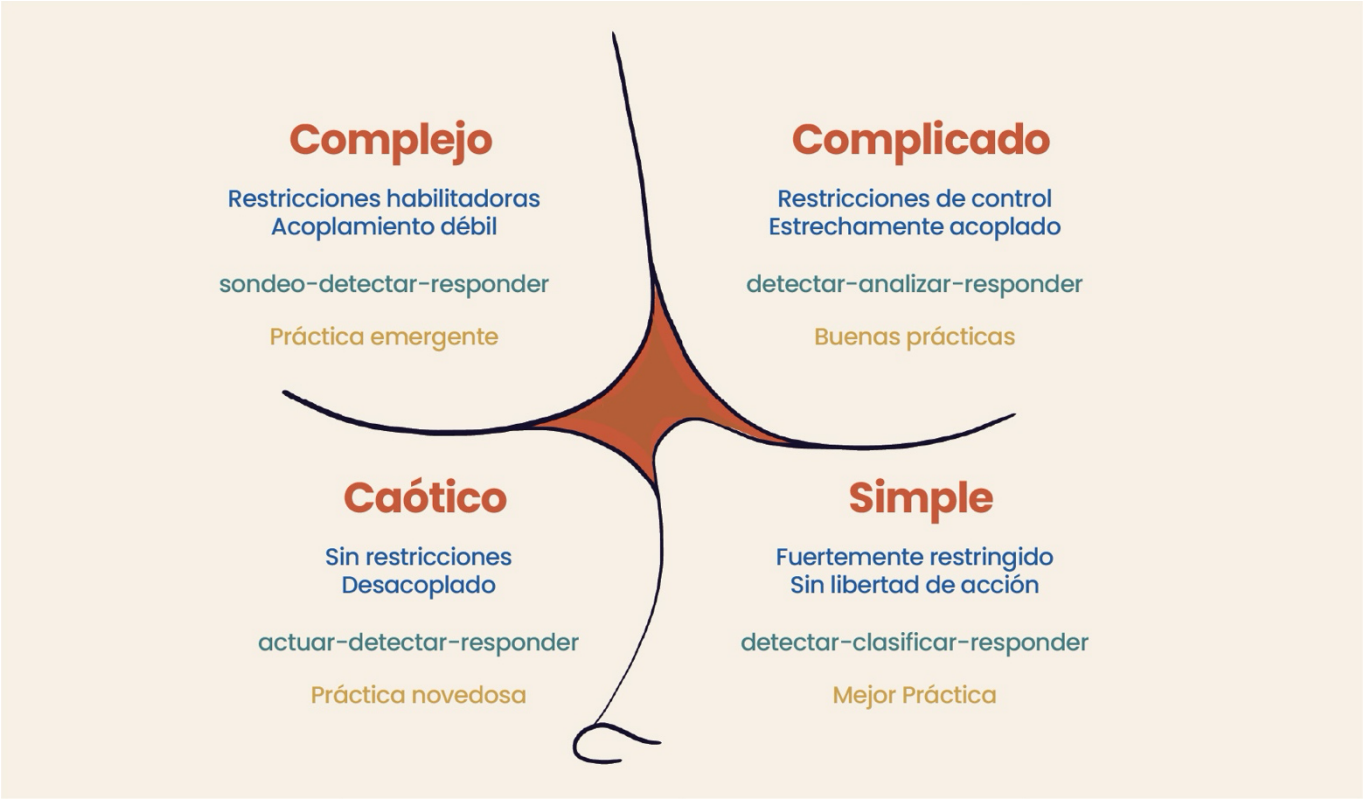
ninguna programación, generaron un sistema de investigación completo que produjo de manera autónoma un informe sustancial con extensas citas trazables. Este es precisamente el salto de solo consumidorxs a constructorxs descrito al inicio de este artículo, no una metáfora, sino un hecho que hemos presenciado de primera mano.

4. Qué construir: un diseño guiado por la naturaleza de la tarea

"Cómo construir" es una pregunta técnica; "qué construir" es una pregunta estratégica. Un error común es este: una vez que se tiene el sistema multiagente declarativo como martillo, toda tarea comienza a parecer un clavo. Tareas de naturaleza fundamentalmente distinta requieren diseños de sistema fundamentalmente distintos. La búsqueda ciega de la automatización total es la trampa más común y más peligrosa.

4.1 El marco Cynefin: comprender la naturaleza de las tareas

El marco Cynefin de Dave Snowden ofrece una herramienta de decisión para juzgar "qué tipo de MAS conviene a esta tarea". Clasifica las tareas según la claridad de sus relaciones causales en cuatro dominios:



Marco Cynefin

El dominio Simple —las relaciones causales son transparentes y el camino es conocido—. Los ejemplos incluyen descargas masivas de archivos, conversiones de formato y limpieza de datos. Estas tareas son adecuadas para flujos de trabajo completamente automatizados que no requieren intervención humana.

El dominio Complicado —las relaciones causales existen pero requieren juicio experto, y los marcos analíticos pueden determinarse de antemano—. Un ejemplo es realizar evaluaciones de países según un sistema de indicadores establecido. Estas tareas son adecuadas para flujos de trabajo analíticos paralelizables: múltiples agentes recopilan y analizan datos simultáneamente a lo largo de distintas dimensiones, y luego consolidan los resultados.

El dominio Complejo —las relaciones causales solo pueden identificarse retrospectivamente, y los conocimientos críticos deben provenir de humanos—. Un ejemplo es realizar un análisis político-económico de la estructura de clases de un país determinado: sin comprender el contexto histórico de las dinámicas raciales, el análisis de datos por sí solo se equivocará por completo. Estas tareas exigen **colaboración iterativa entre humano y máquina**: la IA realiza investigación exploratoria y preparación de materiales, el humano aporta conocimientos y marcos teóricos, y la IA luego realiza análisis profundo bajo la orientación del marco del humano.

El dominio Caótico —la información requerida no existe en ninguna fuente de datos accesible y debe crearse mediante la acción humana—. Un ejemplo es el trabajo de campo realizado en fábricas y comunidades para recopilar materiales primarios: información que no existe en internet, no tiene registro de archivo y está fuera del alcance de la IA. En estas tareas, la IA solo puede servir como herramienta auxiliar para trabajo mecánico, como transcripción, organización de notas y recuperación de material de contexto.

La mayoría de los proyectos de investigación reales abarcan los cuatro dominios simultáneamente, y reconocer esto es esencial para diseñar un sistema efectivo. Consideremos un estudio sobre por qué la riqueza extractiva no ha producido un desarrollo de base amplia en un país dado. Traducir documentos presupuestarios gubernamentales y limpiar conjuntos de datos fiscales son tareas del dominio Simple: totalmente automatizables. Recopilar y analizar datos de ingresos extractivos a lo largo de dimensiones establecidas (tasas de regalías, recaudación de impuestos corporativos, términos de reparto de producción) es una tarea del dominio Complicado: los agentes pueden trabajar en paralelo por producto básico usando un marco analítico predeterminado. Pero la pregunta analítica central —identificar la configuración específica de fuerzas de clase, relaciones Estado-capital y presiones financieras internacionales que producen ese resultado— pertenece al dominio Complejo: requiere el propio juicio político-económico de la investigadora, algo que la IA no puede suministrar. Y las experiencias de las comunidades desplazadas por operaciones mineras —la ira, las promesas incumplidas, las dinámicas políticas locales— pertenecen al dominio Caótico, accesibles solo mediante trabajo de campo que ningún sistema de IA puede realizar. Un sistema bien diseñado automatiza los dos primeros dominios, estructura una colaboración iterativa humano-máquina para el tercero y limita el papel de la IA en el cuarto a la transcripción y la recuperación de contexto.

Un simple "árbol de decisión de tres preguntas" puede ayudar a hacer la determinación: ¿la información necesaria para completar esta tarea ya existe? (No: dominio Caótico). ¿El camino analítico de la información a las conclusiones es claro? (No: dominio Complejo). ¿Requiere juicio de nivel experto? (Sí: dominio Complicado; No: dominio Simple).

4.2 Degradación de dominio: la evolución del proceso de investigación

El conocimiento más profundo que revela el marco Cynefin es este: **el propio proceso de investigación es una progresión desde lo Caótico/Complejo hacia lo Complicado/Simple.**

Un proyecto de investigación típicamente comienza en lo Caótico o Complejo: hay incertidumbre sobre qué estudiar, qué marco emplear o por dónde empezar. A medida que se profundiza la comprensión, la incorporación de marcos humanos degrada las tareas Complejas a Complicadas: una vez que se establecen las dimensiones analíticas y los criterios de evaluación, lo que queda es recopilación y análisis estructurado de datos, algo que la IA puede realizar de manera eficiente. A medida que maduran los métodos, las tareas Complicadas se degradan aún más hacia lo Simple: los procesos de evaluación se codifican y pueden repetirse de manera totalmente automatizada.

Un sistema multiagente bien diseñado debería sustentar esta evolución, incrementando progresivamente el grado de automatización a medida que se profundiza la comprensión. Esta es también la advertencia de diseño más importante: **no forzar la automatización de toda tarea.** Forzar la automatización en tareas de los dominios Complejo o Caótico

habitualmente produce grandes volúmenes de contenido superficial y sin profundidad analítica. Reconocer los límites de la IA es tan importante como reconocer sus capacidades.

4.3 La práctica de GSI: configuraciones distintas para tareas distintas

Los sistemas reales de GSI encarnan configuraciones distintas para cada uno de estos cuatro dominios.

Dominio Simple: el flujo de trabajo totalmente automatizado. Una investigadora tiene un lote de documentos en un idioma que no puede leer y necesita traducirlos a uno que sí conoce. Esta tarea antes requería traductores profesionales; ahora puede confiarse por completo a la IA: las entradas están definidas, las reglas de procesamiento están estandarizadas, el formato de salida es claro. Las características de estas tareas son: las ubicaciones de la información son conocidas, las rutas de recuperación son claras, los métodos de procesamiento están estandarizados; pueden confiarse por completo a la IA para su finalización automatizada.

Dominio Complicado: el flujo de trabajo analítico paralelizable. La evaluación del Índice de Soberanía Digital (DSI, por sus siglas en inglés) es el caso representativo de este dominio. El marco de evaluación abarca 4 dimensiones y 16 indicadores, desarrollados en miles de criterios de evaluación. Para cada país, el sistema recopila cientos de piezas de evidencia sin procesar, que se filtran mediante deduplicación y controles de calidad para producir elementos de alta calidad. El flujo de trabajo completo produce un informe de evaluación estructurado en cuestión de horas: se han completado evaluaciones para los once países de los BRICS. Las características de estas tareas son: el marco analítico ya está establecido (diseñado de antemano por expertos humanos), y lo que resta es recopilación y análisis de datos estructurados a gran escala, algo que la IA puede realizar de manera eficiente en paralelo.

Dominio Complejo: colaboración iterativa humano-máquina. Los estudios de coyuntura de país del RPS representan este dominio. El sistema emplea un marco analítico universal de 19 dimensiones (que abarca los fundamentos materiales, las fuerzas de clase y el análisis de contradicciones), pero al aplicar este marco a un país específico, el paso crítico es la "contextualización": identificar las contradicciones principales específicas de ese país, algo que la IA no puede lograr automáticamente. Tomemos como ejemplo a Sudáfrica: sin comprender el legado histórico del colonialismo de asentamiento, sin comprender cómo las dinámicas raciales han moldeado la forma particular de las contradicciones de clase en el país, la IA puede recopilar toda la información que quiera y aun así perderse por completo el punto esencial. Estas tareas exigen colaboración humano-máquina: la IA procesa miles de fuentes de material y produce docenas de informes de investigación temáticos; el humano, sobre esta base, aporta juicios analíticos críticos (como determinar que Sudáfrica requiere agregar "el legado del colonialismo de asentamiento" como dimensión analítica específica del país); la IA luego, bajo la orientación del marco del humano, entreteje la investigación fragmentada en una narrativa analítica coherente. La incorporación del marco humano es precisamente el paso que degrada una tarea Compleja a una Complicada.

Dominio Caótico: la IA como herramienta auxiliar. El trabajo de campo realizado en fábricas y comunidades para recopilar materiales primarios —información que no existe en internet, no tiene registro de archivo y está fuera del alcance de la IA—. En este dominio, el humano lidera todo el proceso, y la IA solo realiza trabajo auxiliar mecánico como transcripción de audio, organización de notas y recuperación de material de contexto. Aun así, incluso aquí, la asistencia de la IA tiene valor: estudiantes de doctorado, después de usar IA para procesar grabaciones de entrevistas, "descubrieron en las transcripciones información que no habían escuchado ni notado durante las entrevistas mismas".

5. Implicaciones para el Sur Global: la convergencia del MAS declarativo y AI4SS

Las secciones anteriores abordaron el martillo mismo —por qué elegimos este martillo, cómo forjarlo, qué produce en la práctica y qué tipo de clavos impulsa—. Esta sección debe responder una pregunta distinta: **¿por qué importa este martillo especialmente para el Sur Global?**

La respuesta no radica en la tecnología en sí, sino en la alineación estructural entre la tecnología y las condiciones concretas que enfrenta el Sur Global.

5.1 Los obstáculos estructurales que enfrenta el Sur Global

Antes de discutir el potencial de la IA, debemos primero confrontar los obstáculos estructurales que enfrenta el Sur Global.

La asimetría de clase en la producción de conocimiento. Los académicos que sirven a los intereses del capital poseen abundantes recursos, tiempo y energía para producir conocimiento, pero este conocimiento no sirve a los intereses del pueblo, sino que funciona como instrumento de explotación y opresión. Al mismo tiempo, los cuadros de los movimientos populares carecen tanto de recursos como de tiempo. No pueden dedicar horas extensas a la lectura, la investigación y la escritura. La producción y el consumo de conocimiento están monopolizados por líneas de clase: una cruel ironía.

La contradicción entre volumen de información y capacidad cognitiva. Lenin exigió en *Las tareas de las Juventudes Comunistas* (1920) que los comunistas enriquecieran indefectiblemente la memoria con los conocimientos de todas las riquezas creadas por la humanidad, y el método marxista de la totalidad requiere una comprensión integrada que abarque la economía, la política, la cultura y la ideología. Pero estas riquezas de conocimiento son demasiado vastas en cantidad, demasiado rápidos en su cambio y demasiado especializados en sus subdivisiones, excediendo con creces los límites de la cognición individual. El método de la totalidad se convierte así en un ideal que no puede llevarse a la práctica.

Barreras lingüísticas y barreras de recursos. El mayor volumen de conocimiento se produce en inglés, y los recursos de traducción son agudamente escasos. Los recursos y bases de datos académicos cobran tarifas exorbitantes, y los medios progresistas carecen tanto de recursos como de personal. Para una investigadora del Sur Global, acceder a los datos y la literatura que un académico del Norte Global da por sentados a menudo requiere un gasto desproporcionado de tiempo y dinero.

La asimetría de infraestructura Norte-Sur. Los sistemas de IA, las instituciones editoriales y los índices de citas están concentrados en el Norte. Las investigadoras del Sur Global son evaluadas según estándares del Norte, publican en espacios del Norte y ven su trabajo enmarcado por las preocupaciones teóricas del Norte. La asimetría de la infraestructura digital —desde la capacidad de cómputo hasta la conectividad de red y el suministro eléctrico— significa que la ola de la IA corre el riesgo de ampliar, en lugar de reducir, la brecha Norte-Sur.

5.2 Una triple liberación

Reuniendo el análisis anterior, el sistema multiagente declarativo ofrece al Sur Global una triple liberación estructural.

Del consumo tecnológico a la construcción tecnológica: soberanía epistemológica. El MAS declarativo combinado con un lenguaje de patrones como POMASA permite al Sur Global construir sus propias herramientas usando su propio conocimiento de dominio, sin esperar la benevolencia de Silicon Valley. En un sistema declarativo, el plano es el código fuente y la metodología es el programa, y lo que posee el Sur Global es precisamente su propia metodología y conocimiento de dominio. “Publicar un lenguaje de patrones equivale a publicar código fuente” significa una nueva forma de código

abierto: no solo usamos las herramientas de código abierto del Norte; estamos creando los propios activos de código abierto del Sur.

Del monopolio del conocimiento a la democratización del conocimiento: multiplicación de capacidades. Un pequeño equipo de pocas personas, combinado con un MAS declarativo, puede producir resultados que antes requerían una gran institución de investigación —varios cientos de agentes, miles de citas en un solo informe de país, ganancias de eficiencia de 100 a 200 veces—. Esta multiplicación de capacidades tiene una consecuencia particular para el Sur Global: permite que instituciones progresistas con recursos limitados, departamentos de investigación sindical y pequeños medios independientes realicen investigación rigurosa y basada en evidencia, y producción de contenido. La capacidad de producción de conocimiento, antes monopolizada, se está devolviendo.

Del trabajo mecánico a la creación intelectual: la liberación del ser humano. La IA se hace cargo del trabajo mecánico de escritorio —buscar, organizar, transcribir, análisis preliminar—, liberando a los humanos de la pesada carga del procesamiento de información. Pero el propósito de esta liberación no es el ocio; es permitir que los humanos regresen al terreno, a las bases, a las masas, para hacer lo que solo los humanos pueden hacer: generar conocimientos, señalar el camino a seguir y emitir juicios finales. Lo que Marx llamó la unidad de la teoría y la práctica —la *praxis*— debería completarse como un ciclo cerrado dentro de la misma persona. En nuestra práctica, hemos llegado a reconocer con claridad cada vez mayor: "el pensamiento humano, la experiencia humana, el conocimiento humano: estos son los elementos más esenciales, más luminosos, en todo el proceso de producción de conocimiento y contenido". La IA no pretende reemplazarlos; todo lo contrario, todo el valor de la IA radica en permitir que la mente humana se concentre en esas tareas más esenciales que la IA no puede realizar.

El arco descrito a lo largo de este artículo —desde las limitaciones estructurales de una única conversación con la IA, pasando por la arquitectura modular de los sistemas multiagente, hasta el lenguaje de patrones que guía su construcción— traza un recorrido del consumo a la construcción. Una investigadora que comienza pegando su metodología en una ventana de chat y termina construyendo un sistema de cientos de agentes coordinados no solo ha adoptado una nueva herramienta. Ha recuperado la capacidad de dar forma a sus propios medios de producción de conocimiento. Su marco epistemológico se convierte en datos de referencia; su metodología se convierte en un plano de agente; sus estándares de calidad se convierten en puertas de calidad incorporadas. Ella no programa; describe. Y lo que describe no es un deseo, sino una especificación —fundamentada en su formación, su conocimiento de dominio y su compromiso político con las comunidades cuyas condiciones estudia—. El sistema que construye pertenece a su institución; la metodología que codifica es la suya propia; el conocimiento que acumula se compone de proyecto en proyecto.

5.3 Restitución: de «usuarios» a creadores

El significado del MAS declarativo se extiende más allá de las ganancias de eficiencia o la multiplicación de capacidades: toca un proceso histórico más profundo.

Durante las últimas cuatro décadas, la industria de la tecnología de la información ha transformado sistemáticamente a la vasta mayoría de las personas de creadoras de herramientas en meras "usuarias". Como documentó Bonnie Nardi en *A Small Matter of Programming* [Una pequeña cuestión de programación], la interfaz gráfica de usuario rompió la continuidad entre usar y crear; la mercantilización del software cerró los caminos hacia la construcción; las comunidades de desarrolladores evolucionaron hacia gremios de conocimiento; la complejidad cada vez más creciente elevó la barrera cada vez más alto. Al final, no solo desaparecieron las habilidades: el concepto mismo de "puedo construir herramientas para mí misma" se desvaneció del lenguaje y la imaginación de las personas. Cuando algo no puede decirse, no puede pensarse. La desposesión de capacidad se volvió así permanente y autosustentada.

El sistema multiagente declarativo revierte esta corriente subterránea. En esta arquitectura, los planos son Markdown en lugar de código; el conocimiento de dominio es el programa; la metodología es el software. Construir un sistema multiagente capaz de investigación profunda autónoma ya no requiere el permiso del gremio de programadores: lo que requiere es el conocimiento en la mente de la experta de dominio: a lo largo de qué dimensiones analizar un problema, de qué fuentes recopilar información, qué marco epistemológico debe guiar el juicio. Y este conocimiento es precisamente lo que las fuerzas progresistas del Sur Global nunca han tenido en falta.

Esto no es una dotación; es una restitución.

La capacidad de las personas comunes para construir sus propias herramientas de información existió desde el principio: las investigadoras de la era Unix escribían sus propios scripts; las usuarias de la era de la computadora personal encendían sus máquinas hacia una línea de comandos BASIC. Esta capacidad no es una novedad traída por el progreso tecnológico; es un antiguo derecho que fue arrebatado sistemáticamente por cuatro décadas de evolución industrial. El sistema multiagente declarativo lo devuelve.