Deep Learning para Business Analytics

Día 14 · Tema 5 — Multi-agente y MCP

Orquestador-trabajador, herramientas conectadas.

Eduardo C. Garrido-Merchán

Profesor Propio Adjunto, Métodos Cuantitativos, ICADE

ecgarrido@comillas.edu

Madrid, Día 14

ejemplos del día

Lo que vamos a cubrir hoy.

1. Subagentes y orquestación

2. Model Context Protocol (MCP)

3. Human-in-the-loop (HITL)

4. Caso comercial completo

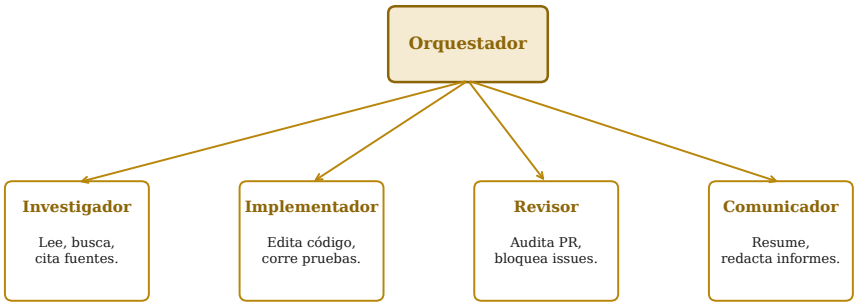
Al final del día: sabes diseñar un sistema multi-agente, justificar el uso de MCP en una empresa y elegir el nivel de human-in-the-loop.

Parte 1 / 4

Subagentes y orquestación

Patrón orquestador-trabajador 🤖.

Patrón orquestador-trabajador: un agente reparte el trabajo, varios lo ejecutan.



Por qué especializar subagentes 🤖.

Foco. Cada subagente lleva un `CLAUDE.md` más estricto que el del orquestador. Saber poco mejor que saber mal.

Herramientas restringidas 🔧. Un *Implementador* no tiene WebSearch; un *Investigador* no tiene Edit ni Bash. Limita la superficie de error.

Paralelismo. Tres subagentes pueden trabajar en paralelo en archivos independientes. La latencia total cae.

Auditoría 🛡️. La traza queda más limpia: sabes qué subagente decidió qué.

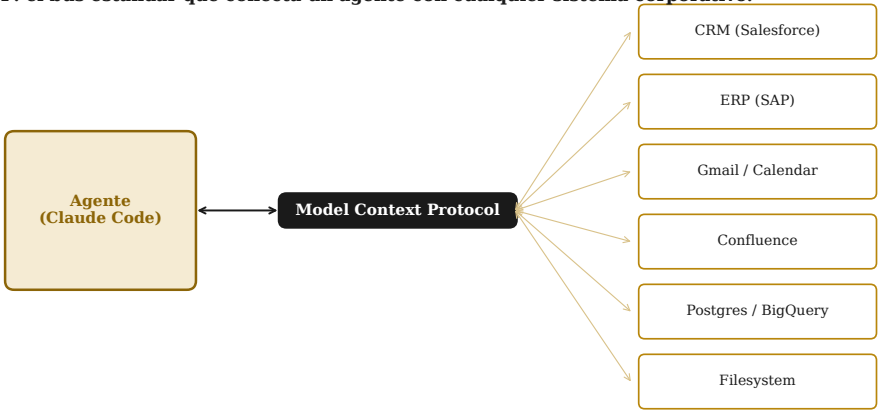
Idea clave. La especialización es la única forma realista de escalar agentes en proyectos serios.

Parte 2 / 4

Model Context Protocol (MCP)

MCP 🔌: el bus que conecta agentes con cualquier sistema.

MCP: el bus estándar que conecta un agente con cualquier sistema corporativo.



Tool use 🔧 a bajo nivel: JSON estructurado.

Tool use: el modelo se comunica con el mundo en JSON.

El modelo emite tool_call:

```
{"name": "search_crm",
 "args": {"customer_id": "C-1234"}}
```

→

El servidor MCP devuelve:

```
{"name": "Acme S.A.",
 "tier": "gold", "open_tickets": 2}
```

El agente lee el resultado, decide qué hacer y puede llamar a más tools en cadena.

Buena práctica: cada tool define entradas, salidas y errores en JSON Schema.

Esto convierte un agente en una pieza testeable, no en una caja negra.

Concepto → aplicación: qué servidor MCP 🔌 para qué caso.

Servidor MCP	Caso de negocio	Datos accesibles	Riesgo
Filesystem	Asistente de código local.	Archivos locales.	Bajo.
Postgres	Analista interno (Mercadona).	Tablas vending.	Medio (PII).
Salesforce	Asistente comercial 🛒 (Telefónica).	Leads, contactos.	Medio.
Gmail / Calendar	Asistente personal corporativo.	Mails, eventos.	Alto (privacidad).
GitHub	Revisor de PR (BBVA).	Repos, issues.	Bajo-medio.

Idea clave. Cada servidor MCP empotrado en el agente añade superficie de ataque; el gobierno (Tema 7) la limita.

Parte 3 / 4

Human-in-the-loop (HITL)

Cinco niveles de supervisión humana 🛡️.

Cinco niveles de human-in-the-loop, de más vigilancia a más autonomía.

Nivel	Descripción	Riesgo	Caso
0	Humano hace todo, IA solo asiste.	Bajo	Asistente que sugiere.
1	Agente propone; humano aprueba paso a paso.	Bajo-medio	Refactor con /plan.
2	Agente actúa; humano revisa al final del trabajo.	Medio	Triage de tickets.
3	Agente actúa con supervisión por muestreo.	Alto	Asistente comercial.
4	Agente totalmente autónomo, sin revisión humana.	Crítico	Solo investigación interna.

Cómo elegir el nivel según el caso.

Regla 1 — Coste del error 💰. Si una decisión equivocada cuesta más de 10 $, exige humano (nivel 1 o 2).

Regla 2 — Velocidad. Si la decisión necesita estar tomada en < 2 minutos, nivel 3 o superior, con muestreo.

Regla 3 — Frecuencia. Si la tarea se repite miles de veces al día, nivel 2 con auditoría por muestreo.

Regla 4 — Trazabilidad regulatoria 🛡️. Para sectores regulados (banca, salud, seguros), nunca nivel 4. La AI Act lo exige (Tema 7).

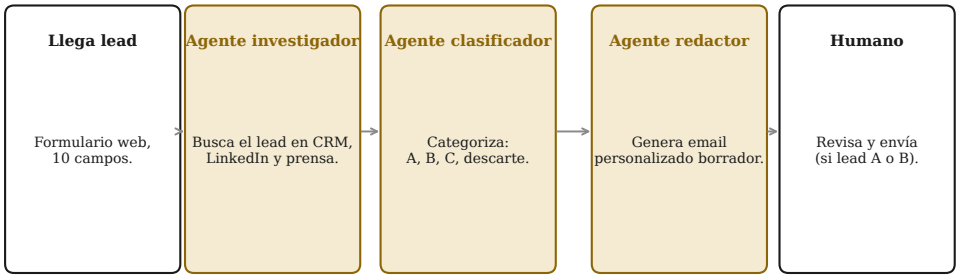
IDEA CLAVE. El nivel HITL es una decisión de *gobierno*, no técnica.

Parte 4 / 4

Caso comercial completo

Multi-agente 🤖: asistente comercial 🛒 de BBVA.

Ejemplo multi-agente: asistente comercial de BBVA, 3 subagentes + humano.



Cálculo a mano: cuánto ahorra 💰 el sistema multi-agente.

Lead manual. Un comercial tarda 25 minutos por lead (CRM + LinkedIn + redacción). 12 leads/día por comercial.

Con sistema multi-agente 🤖. El comercial revisa borradores. Tarda 4 minutos por lead. 70 leads/día por comercial.

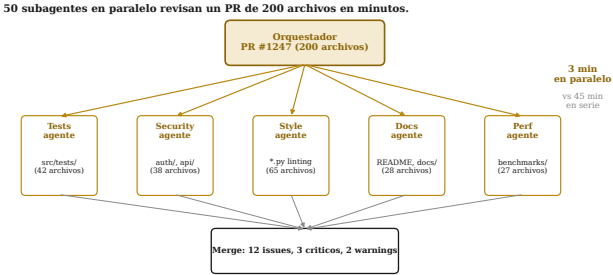
Productividad. factor $= 70/12 \approx 5{,}8\times$.

Coste del sistema. 0,02 $ por lead procesado (3 llamadas LLM × 0,007 $). Para 70 leads/día son 1,40 $/día y 30 $/mes por comercial.

Reducción del coste por lead. De 12,50 $ (25 min × 30 $/h) a 2,02 $ (4 min × 30 $/h + 0,02 $).

Idea clave. ROI claro: cualquier banco grande recupera la inversión en semanas.

50 subagentes en paralelo revisan un PR de 200 archivos 🤖.



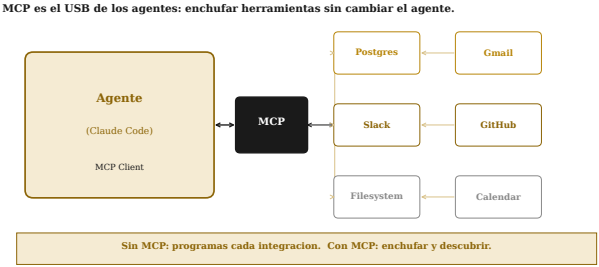
Problema 🕸️**.** Un solo agente se pierde en 200 archivos. El contexto desborda y la calidad cae.

Solucion. Dividir y conquistar. El orquestador reparte archivos a subagentes especializados: tests, security, style, docs, rendimiento.

Resultado. 3 minutos en paralelo frente a 45 minutos en serie. Cada subagente reporta issues al orquestador, que fusiona.

Idea clave. La especializacion es la unica forma realista de escalar agentes en proyectos serios.

MCP es el USB de los agentes: enchufar sin cambiar el agente 🔌.



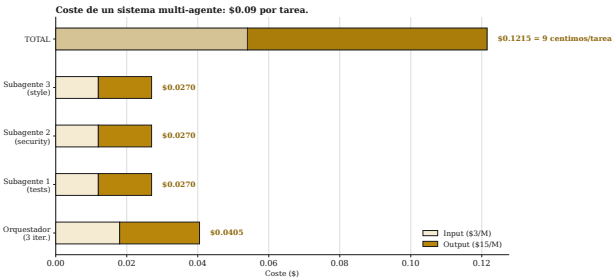
Analogia 🔧. Un portatil tiene puertos USB. Un agente tiene puertos MCP. Enchufas una base de datos, un correo o un calendario, y funciona.

Sin MCP. Hay que programar cada integracion a mano: parsear APIs, manejar auth, traducir formatos.

Con MCP. El agente descubre automaticamente las herramientas disponibles y sus esquemas JSON.

Idea clave. MCP es el estandar emergente. Aprenderlo ahora es invertir en la plataforma correcta.

Coste de un sistema multi-agente: orquestador + 3 subagentes 💰.



Calculo. Orquestador: 2.000 tokens in + 500 out por iteracion, 3 iteraciones. Cada subagente: 4.000 in + 1.000 out, 1 llamada.

Tokens totales.
$3 \times (2000+500) + 3 \times (4000+1000)$
$= 7{,}500 + 15{,}000 = 22{,}500.$

A $3/M in, $15/M out. Input: $0,054. Output: $0,0675. Total: $0,12 por tarea.

Idea clave. Un sistema multi-agente completo cuesta unos 12 centimos por tarea.

HITL: cinco niveles de autonomia, del confirma-todo al avisame-si-fallas 🛡️.

Regla ⚖️. Ajustar el nivel al riesgo de la tarea, no al entusiasmo del equipo.

Nivel 1. Aprobar cada accion. Para deployments a produccion.

Nivel 2-3. Aprobar solo acciones destructivas o externas. Para refactoring y triage.

Nivel 4-5. Actuar libremente. Solo para investigacion interna o sandboxes.

IDEA CLAVE. El nivel HITL es una decision de gobierno, no tecnica. Depende de coste, velocidad y regulacion.

Lo que el alumno se lleva a casa de este día.

Concepto 1. Sistemas multi-agente 🤖 con orquestador y trabajadores escalan donde un agente único se atasca.

Concepto 2. MCP 🔌 es el estándar emergente para conectar agentes con CRMs, ERPs, mail, calendarios.

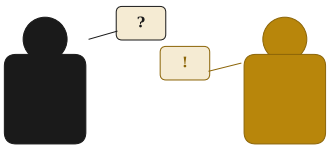
Concepto 3. Tool use 🔧 en JSON estructurado convierte un agente en una pieza testeable.

Concepto 4. El nivel de HITL 🛡️ es una decisión de gobierno: depende de coste, velocidad y regulación.

Para la práctica. Configurar un servidor MCP de Filesystem y darle al agente acceso a una carpeta concreta.

Próximo día. Día 15 (Tema 6.1): RAG corporativo.

Pausa para debatir: ¿Quién es responsable si un sistema multi-agente daña?



La pregunta. Un orquestador con cuatro subagentes decide algo dañino. Ningún humano aprobó la decisión. ¿Quién responde?

Posición A. El proveedor del modelo: vendió la herramienta.

Posición B. La empresa: integró, desplegó y supervisó el sistema.

Reglas. 5 minutos, dos turnos de palabra por bando, móviles boca abajo. Yo modero, no participo.

Hasta el Día 15.

RAG: conocimiento privado, citas verificables.

Eduardo C. Garrido-Merchán

ecgarrido@comillas.edu