MANUAL · CLASE 6

Secretaría de Extensión Universitaria

CURSOS

UTN RECONQUISTA

Cómo funcionan los Subagentes de IA

Arquitectura multiagente, orquestación y cómo se usan para sistematizar y automatizar procesos

Capacitación «Gestión de Costos de Obra + IA»

DOCENTE: EMILSE KASEM

CONTENIDO

AGENDA

· Introducción — de un agente a un equipo de agentes

1 Qué es un subagente de IA

2 Por qué dividir el trabajo en subagentes

3 Cómo se arma un subagente: arquitectura

4 Patrones de orquestación multiagente

5 Cómo se comunican los subagentes entre sí

6 Cómo sistematizan y automatizan procesos

7 Cuándo conviene usar subagentes

8 Cuándo NO conviene: el caso del agente único

9 Limitaciones y riesgos específicos

10 Buenas prácticas de diseño

11 Caso aplicado: sistematizar un proceso

G Glosario de términos clave

S Síntesis final

DE UN AGENTE A UN EQUIPO DE AGENTES

Qué es este manual

Este manual es la continuación natural de «Cómo funciona un agente de IA»: mientras aquel explicaba la arquitectura de un agente individual, este se enfoca en qué pasa cuando varios agentes trabajan juntos. A esos agentes especializados que un agente principal convoca para resolver una parte acotada de un problema se los llama **subagentes**, y el conjunto forma un **sistema multiagente**.

El interés práctico no es solo técnico: entender cómo funcionan los subagentes es entender cómo se usa la IA hoy para **sistematizar** procesos —convertir una tarea que depende de la memoria y el criterio de una persona en una secuencia explícita y repetible de pasos— y para **automatizarlos**, haciendo que esa secuencia corra sola.

Cómo está organizado

El manual recorre, en once capítulos, qué es un subagente y en qué se diferencia de un agente único; por qué conviene dividir el trabajo en piezas más chicas; cómo se arma un subagente y cómo lo invoca un orquestador; los patrones de coordinación multiagente más usados en 2026; cómo se comunican los subagentes entre sí; cómo ese diseño se traduce en sistematización y automatización real; en qué casos los subagentes aportan una ventaja medible y en cuáles un agente único sigue siendo la mejor opción; las limitaciones y riesgos propios de estos sistemas; y un capítulo de buenas prácticas, cerrando con un caso aplicado completo, un glosario y una síntesis.

1.1 DEFINICIÓN

Subagente

Un subagente es una instancia de un agente de IA a la que otro agente —llamado **orquestador** o agente principal— le delega una tarea acotada y específica, para que la resuelva de forma relativamente independiente y devuelva un resultado. A diferencia de una simple llamada a una función, el subagente no ejecuta un procedimiento fijo: recibe un objetivo en lenguaje natural, razona sobre cómo cumplirlo, puede usar sus propias herramientas y, cuando termina, entrega su resultado al agente que lo convocó.

Lo distintivo

Opera en su propia ventana de contexto, separada de la del orquestador: tiene su propio prompt de sistema, sus propias herramientas permitidas y, en muchos casos, su propio modelo de IA asignado según la complejidad de la tarea.

1.2 DE UN EMPLEADO POLIVALENTE A UN EQUIPO CON ROLES

Agente único — el empleado polivalente

Es como una persona que investiga, redacta, revisa y archiva por su cuenta: funciona, pero a medida que la tarea crece, esa misma persona tiene que sostener en la cabeza cada vez más información, y es más fácil que pierda el hilo o mezcle contextos.

Sistema de subagentes — el equipo con roles

Cada integrante tiene un rol acotado y la información justa para cumplirlo, y hay alguien —el orquestador— que reparte las tareas, hace seguimiento y junta los resultados al final. Ningún integrante necesita saber todo lo que hacen los demás; solo hacer bien su parte y entregarla.

1.3 EL PATRÓN DE FONDO: ORQUESTADOR CON SUBAGENTES AISLADOS

Convergencia de la industria en 2026

Durante 2026, buena parte de la industria —Anthropic, OpenAI, Cognition y otros desarrolladores de agentes— convergió hacia un mismo patrón: un orquestador que conserva el contexto completo de la tarea y convoca «trabajadores» temporales (los subagentes), cada uno enfocado en una porción del problema, que devuelven solo un resumen de lo que hicieron, no la totalidad de su proceso interno.

Este patrón se impuso frente a diseños alternativos donde varios agentes «pares» colaboran sin una jerarquía clara, que en la práctica resultaron más difíciles de coordinar y de auditar (Capítulo 8).

Ejemplo aplicado: subagentes en la práctica

Cuando un agente principal necesita revisar miles de líneas de código para encontrar el origen de un error, puede delegar esa búsqueda a un subagente especializado. Ese subagente puede generar cientos de líneas de resultados intermedios —registros de error, búsquedas, intentos fallidos—, pero al orquestador solo le llega el resultado final: «el error está en el archivo X, línea Y, causado por Z». Así el agente principal sigue trabajando con un contexto liviano y enfocado.

CINCO RAZONES PARA DIVIDIR EL TRABAJO

Dividir una tarea grande entre varios subagentes en lugar de resolverla con un único agente no es una moda técnica: responde a cinco razones concretas, cada una con un beneficio medible.

1

Aislamiento de contexto

Todo el «ruido» de una tarea larga queda contenido dentro del subagente y nunca satura el contexto del agente principal.

2

Especialización y foco

Un rol acotado necesita instrucciones más simples y en la práctica se comporta de forma más confiable que un generalista.

3

Paralelización

Varias tareas independientes al mismo tiempo; el tiempo total queda limitado por la rama más lenta, no por la suma de todas.

4

Seguridad y permisos acotados

Cada subagente accede solo a las herramientas y datos que necesita, separando dominios sensibles de forma auditable.

5

Reutilización

Un subagente bien definido se reutiliza en distintos flujos, como una función bien escrita en un programa.

3.1 LOS COMPONENTES DE UN SUBAGENTE

Componente	Qué es
Un rol y un nombre	Qué hace este subagente y cuándo corresponde convocarlo.
Un prompt de sistema propio	Un manual de instrucciones acotado, separado del prompt del orquestador.
Una lista de herramientas permitidas	Solo las habilitadas explícitamente, nunca el conjunto completo del sistema.
Un modelo asignado	Uno más liviano para tareas simples, uno más potente para tareas que requieren mayor razonamiento.
Permisos independientes	Qué puede leer, modificar o ejecutar, configurado por separado del orquestador y de otros subagentes.

3.2 Y 3.3 INVOCACIÓN Y RETORNO

3.2 · Cómo lo invoca el orquestador

El agente principal puede delegar trabajo de distintas formas: de manera **automática**, cuando decide por su cuenta que una parte de la tarea encaja con la descripción de un subagente disponible; de manera **explícita**, cuando se lo nombra directamente; en **primer plano**, quedando bloqueado hasta que el subagente termina; o en **segundo plano**, dejando que trabaje de forma concurrente mientras el orquestador sigue avanzando con otras partes de la tarea.

3.3 · Qué vuelve al orquestador

Este es el punto de diseño más importante de toda la arquitectura: lo que regresa al orquestador no es la transcripción completa de todo lo que hizo el subagente, sino un resumen comprimido con el resultado final y la información que efectivamente necesita para seguir adelante.

Ese filtro mantiene manejable el contexto del agente principal incluso en tareas muy largas, y exige diseñar con cuidado qué información es imprescindible que llegue y cuál puede quedar descartada dentro del subagente.

PATRONES DE ORQUESTACIÓN (1/2)

No todos los sistemas de subagentes se coordinan de la misma manera. En 2026 conviven cinco patrones principales, cada uno con sus propias ventajas y costos.

Patrón	Cómo funciona	Ventaja	Riesgo o costo
Supervisor (delegación jerárquica) — el patrón por defecto en 2026	Un orquestador descompone la tarea en subtareas que no se superponen, delega cada una a un subagente especializado y después integra los resultados.	Mejor soporte nativo en los principales frameworks; sus fallos son bien conocidos y relativamente fáciles de prevenir.	Bucles de sobre-delegación si no se define un límite claro de iteraciones.
Fan-out (paralelo: dispersar y recolectar)	Reparte la misma tarea —o varias subtareas distintas— entre múltiples subagentes que trabajan en simultáneo y junta los resultados cuando todas las ramas terminan. Ideal para investigación en paralelo, revisión concurrente de documentos o enriquecimiento de datos desde múltiples orígenes.	El tiempo total queda acotado por la rama más lenta.	Hay que definir de antemano qué hacer si una rama falla.
Pipeline (cadena secuencial)	Cada subagente recibe como entrada el resultado del anterior, en secuencia estricta: investigar → redactar un borrador → criticarlo → revisarlo.	Es la topología más simple de coordinar: cada etapa puede ver el contexto de la anterior.	Falla en cascada: si un resultado intermedio sale mal, contamina todo lo que viene después.

PATRONES DE ORQUESTACIÓN (2/2)

Patrón	Cómo funciona	Ventaja	Riesgo o costo
Debate (crítica multi-perspectiva)	Varios subagentes abordan la misma consigna de forma independiente y después se critican entre sí, o un agente «juez» arbitra entre sus respuestas.	Mejora la calidad de las decisiones de alto impacto porque incorpora perspectivas divergentes.	Del orden de 2,5 veces el costo de usar un único modelo: reservarlo para decisiones críticas, no para tareas rutinarias.
Swarm (coordinación dinámica de pares)	Una población abierta de agentes «pares» se activa dinámicamente según la carga de trabajo, coordinándose por memoria compartida o mensajes en lugar de una jerarquía fija.	Permite escalar a decenas o cientos de agentes en sesiones autónomas muy largas.	Infraestructura considerable, costo variable difícil de acotar y mecanismos explícitos contra condiciones de carrera. Hoy es el patrón menos maduro.

Cómo elegir el patrón. La recomendación general es empezar siempre por el patrón supervisor —el más simple, el más predecible y el mejor soportado— y sumar complejidad (fan-out, debate o swarm) solo cuando el volumen o la naturaleza del trabajo realmente lo justifique. Migrar directamente a un patrón swarm para una tarea que un supervisor simple resolvería igual de bien solo agrega costo y superficie de fallo.

CÓMO SE COMUNICAN LOS SUBAGENTES

5.1 · El brief estructurado

El patrón de comunicación más usado en 2026 no es una conversación abierta entre agentes, sino un **«brief»** estructurado: cada subagente recibe un prompt de sistema propio que define su rol, más un primer mensaje que especifica el objetivo concreto de la tarea, el formato de salida esperado, qué herramientas puede o conviene usar, y los límites explícitos de lo que debe y no debe hacer. Cuanto más preciso ese brief, más confiable resulta el resultado.

5.2 · Aislamiento por defecto

Salvo que se lo habilite de forma explícita, un subagente no comparte memoria de trabajo con otros subagentes ni con el orquestador: solo ve la información que se le entregó puntualmente en su brief.

Este aislamiento es intencional, aunque exige pensar con cuidado qué información necesita realmente cada subagente, porque nada que no esté en el brief va a estar disponible.

5.3 · Resúmenes comprimidos

Lo que un subagente devuelve al orquestador es un resultado resumido, no el detalle completo de cómo llegó a él. Esto permite que el sistema escale sin saturar el contexto del orquestador, pero tiene una contracara: si el resumen omite un dato relevante, esa pérdida puede no detectarse hasta más adelante.

Por eso el formato de salida esperado es una de las partes más importantes del brief.

CÓMO SISTEMATIZAN Y AUTOMATIZAN PROCESOS

6.1 · Sistematizar: de lo implícito a lo explícito

Sistematizar un proceso significa tomar algo que hoy funciona de forma implícita —una secuencia de pasos que una persona conoce de memoria, o repartida de manera informal entre distintas personas y herramientas— y convertirlo en una secuencia explícita, documentada y repetible de pasos, roles y reglas. Un sistema de subagentes es, en un sentido muy literal, esa sistematización llevada a software: cada paso del proceso se convierte en el rol acotado de un subagente específico.

Esto obliga a que el proceso quede explícito antes de automatizarlo: no se puede convertir en subagentes un proceso que nadie puede describir paso a paso.

6.2 · Automatizar: que el sistema corra solo

Automatizar es el paso siguiente: una vez sistematizado en un flujo de subagentes, ese flujo puede dispararse solo —por un evento o una tarea programada— sin que una persona opere manualmente cada paso.

La persona deja de ejecutar el proceso y pasa a supervisarlo: revisa resultados, interviene en los puntos de control definidos, y ajusta el sistema cuando el proceso de negocio cambia.

6.3 EJEMPLO APLICADO: ATENCIÓN A NUEVAS CONSULTAS

Proceso original (manual)

Cada vez que llega una consulta nueva por correo o formulario, alguien la lee, busca información del contacto, evalúa si es un lead de calidad, redacta una respuesta y actualiza una planilla de seguimiento. Todo eso vive en la cabeza de esa persona y depende de que esté disponible para hacerlo.

Sistematización

El proceso se descompone en roles claros: un subagente de **triage** que clasifica la consulta según urgencia y calidad; uno de **investigación** que busca información pública del contacto; uno de **redacción** que prepara una respuesta personalizada; y uno de **registro** que actualiza el sistema de seguimiento. Los coordina un orquestador con patrón supervisor.

Automatización

El orquestador se dispara solo cada vez que llega una consulta nueva. Triage e investigación corren en paralelo (fan-out) porque son independientes; sus resultados alimentan a redacción (tramo de pipeline); y registro corre al final.

Para los leads de mayor valor, el sistema deja la respuesta preparada pero espera aprobación humana antes de enviarla (nivel semi-autónomo); las consultas rutinarias se responden y registran sin intervención.

CUÁNDO CONVIENE USAR SUBAGENTES

La evidencia disponible en 2026 es clara en marcar escenarios donde dividir el trabajo en subagentes aporta una ventaja medible frente a un agente único.

7.1 · Trabajo paralelizable y de lectura intensiva

Tareas como investigación independiente sobre múltiples fuentes, enriquecimiento de datos desde distintos orígenes o inteligencia de mercado se benefician claramente de repartirse entre subagentes que trabajan en simultáneo, porque cada rama es independiente de las demás y el resultado final no depende de que se ejecuten en un orden particular.

7.2 · Confiabilidad en dominios acotados

Cuando cada subagente tiene un rol muy específico, su tasa de acierto puede ser sensiblemente mayor que la de un agente generalista enfrentando la misma tarea completa. Un estudio de 2026 sobre respuesta automatizada a incidentes técnicos reportó una tasa de recomendaciones accionables notablemente más alta con subagentes especializados, lo cual ilustra el valor de la especialización donde la precisión importa más que la velocidad.

7.3 · Dominios de herramientas separados

Cuando existen razones de seguridad o gobernanza para que ciertas herramientas o datos no se mezclen —por ejemplo, separar el acceso a facturación del acceso a sistemas de desarrollo—, dividir el trabajo en subagentes con permisos disjuntos es una medida de control de acceso mucho más clara y auditable que depender de que un único agente con acceso amplio «se porte bien».

EL CASO DEL AGENTE ÚNICO

Tan importante como saber cuándo usar subagentes es saber cuándo no conviene: un sistema multiagente no es automáticamente mejor, y en varios escenarios comunes resulta claramente peor.

8.1 · El agente único iguala o supera en razonamiento encadenado

En tareas multi-hop (cada paso depende lógicamente del anterior), la evidencia de 2026 muestra que un agente único iguala o supera a un sistema multiagente con la misma cantidad de tokens de razonamiento. Repartirlo entre subagentes no mejora el resultado y en muchos casos lo empeora, porque fragmenta un razonamiento que se beneficia de un mismo hilo continuo.

8.2 · El costo es sensiblemente mayor

Un sistema multiagente puede consumir del orden de quince veces más tokens que una interacción simple con un único agente, porque cada subagente procesa su propio contexto, su propio prompt de sistema y, en algunos patrones, repite información que ya procesó otro. Ese costo solo se justifica cuando el beneficio supera claramente el costo extra.

8.3 · Fallas específicas de los diseños entre pares

Debate y swarm muestran fallas documentadas y recurrentes: repetición completa de la transcripción cada vez que un agente «despierta», inflación progresiva del prompt de sistema, un sobrecosto de «traducción» del supervisor (hasta la mitad de los costos evitables en algunos análisis), supuestos contradictorios entre agentes en paralelo y consenso prematuro entre pares.

8.4 · La conclusión práctica

Para tareas secuenciales, o que dependen de un estado compartido que todos los pasos necesitan ver completo, un agente único suele rendir igual o mejor con el mismo presupuesto de tokens. Hay que medir antes de multiplicar agentes: probar primero si un agente único no resuelve la tarea igual de bien, y sumar subagentes solo cuando haya una razón concreta que lo justifique.

LIMITACIONES Y RIESGOS ESPECÍFICOS

1

Anidamiento limitado

En muchos frameworks de 2026 un subagente no puede crear sus propios subagentes: la delegación funciona en un único nivel.

2

Fallas en cascada (pipeline)

Un resultado incorrecto en una etapa intermedia se traslada, sin filtro, a todas las etapas siguientes.

3

Bucles de sobre-delegación

Sin un límite explícito de iteraciones, el orquestador puede quedar delegando y re-delegando sin converger.

4

Estado no persistente por defecto

Salvo que se habilite memoria explícita, cada invocación de un subagente empieza desde cero.

5

Costo, latencia y menor previsibilidad

Un sistema de subagentes tiende a costar más y a ser más difícil de predecir en tiempo y consumo que un agente único.

9.6 · Los mismos riesgos del agente individual, multiplicados. Las alucinaciones, la inyección de instrucciones, los sesgos heredados y la falta de trazabilidad descriptos en el primer manual no desaparecen: se multiplican por la cantidad de subagentes, y se suma el riesgo de que un resumen mal filtrado oculte información relevante al pasar de un subagente a otro. Un sistema con cinco subagentes tiene, en principio, cinco veces más puntos donde algo puede salir mal, lo cual refuerza la importancia de la observabilidad y del control humano en los puntos críticos.

BUENAS PRÁCTICAS DE DISEÑO

Empezar por el patrón supervisor

El más simple, el mejor soportado y el de fallos más conocidos; sumar complejidad solo cuando el trabajo lo exija.

Escribir un brief explícito para cada subagente

Objetivo, formato de salida esperado, herramientas sugeridas y límites claros de lo que debe y no debe hacer.

Definir límites de iteración y de costo por tarea

Para evitar bucles de sobre-delegación y acotar de antemano cuánto puede costar o tardar una tarea.

Restringir las herramientas al mínimo necesario

Aplicando el principio de necesidad mínima también entre subagentes dentro del mismo sistema.

Diseñar la estrategia de fallos antes de producción

Qué hacer si una rama de un fan-out falla, o si un paso de un pipeline devuelve algo inválido.

Cuidar el formato del resumen que cada subagente devuelve

Es la única información que el orquestador va a tener; un resumen incompleto propaga el error a todo el proceso.

Medir antes de multiplicar agentes

Probar primero si un agente único no rinde igual o mejor con menos costo y menos superficie de fallo.

Mantener supervisión humana en los puntos de mayor impacto

Los mismos criterios de autonomía progresiva del primer manual aplican a cada subagente del sistema.

INFORME SEMANAL DE AVANCE (1/2)

Cómo se vería sistematizar y automatizar, de punta a punta, la preparación de un informe semanal de avance de proyecto usando un sistema de subagentes.

Proceso original (manual)

Cada semana, alguien revisa el estado de las tareas en la herramienta de gestión de proyectos, lee los últimos correos e intercambios relevantes, identifica desvíos respecto del cronograma, y redacta un informe de una página con el estado general, los riesgos detectados y los próximos pasos.

Paso 1 — Sistematización

El proceso se descompone en tres roles. Un subagente de **relevamiento** consulta la herramienta de gestión de proyectos y extrae el estado de cada tarea contra el cronograma. Un subagente de **análisis de riesgo** revisa esos datos junto con los correos de la semana para identificar desvíos. Un subagente de **redacción** arma el informe final en el formato habitual.

INFORME SEMANAL DE AVANCE (2/2)

Paso 2 — Orquestación

El subagente de relevamiento corre primero, porque el de análisis de riesgo necesita sus datos. Por tratarse de un informe donde el orden de la información importa, el equipo optó por un patrón **pipeline** simple: relevamiento → análisis de riesgo → redacción.

Paso 3 — Automatización

El proceso completo se programa para dispararse solo, automáticamente, todos los viernes a la mañana.

Paso 4 — Supervisión

El informe no se envía automáticamente al equipo ni a los clientes: queda como borrador a la espera de que una persona lo revise y apruebe antes de distribuirlo (nivel semi-autónomo, coherente con un documento que sale hacia afuera del equipo), mientras que la extracción de datos y el análisis interno corren de punta a punta sin intervención humana.

Resultado: *lo que antes dependía de que una persona se sentara media hora todos los viernes a juntar información dispersa, ahora es un sistema explícito, documentado y repetible, que además deja un registro auditable de qué datos usó y qué análisis hizo en cada corrida.*

GLOSARIO

TÉRMINOS CLAVE (1/2)

Subagente. Instancia de un agente de IA a la que otro agente (el orquestador) delega una tarea acotada, con su propio contexto, prompt de sistema y herramientas.
Orquestador. El agente que conserva el contexto general de la tarea, decide qué delegar y a quién, y sintetiza los resultados de los subagentes.
Sistema multiagente. El conjunto formado por un orquestador y los subagentes que convoca para resolver una tarea.
Patrón supervisor. El orquestador descompone la tarea en subtareas no superpuestas y las delega a subagentes especializados.

Fan-out. Varias subtareas independientes se reparten entre subagentes que trabajan en paralelo.
Pipeline. Patrón secuencial donde cada subagente recibe como entrada el resultado del anterior.
Debate. Varios subagentes abordan la misma consigna de forma independiente y se critican entre sí o son arbitrados.
Swarm. Coordinación dinámica entre un número abierto de agentes pares, sin jerarquía fija.

TÉRMINOS CLAVE (2/2)

Aislamiento de contexto. Cada subagente opera en su propia ventana de contexto, separada de la del orquestador y de otros subagentes.
Brief estructurado. El prompt de sistema y el mensaje inicial que definen el rol, objetivo, formato de salida y límites de un subagente.
Resumen comprimido. El resultado resumido que un subagente devuelve al orquestador, en lugar de la transcripción completa.
Anidamiento. La posibilidad (o imposibilidad, en la mayoría de los frameworks de 2026) de que un subagente cree sus propios subagentes.

Falla en cascada. Cuando un resultado incorrecto en una etapa de un pipeline contamina todas las etapas siguientes.
Razonamiento multi-hop. Razonamiento encadenado donde cada paso depende lógicamente del anterior.
Sistematizar. Convertir un proceso implícito en una secuencia explícita, documentada y repetible de pasos, roles y reglas.
Automatizar. Hacer que un proceso ya sistematizado corra por sí solo, sin operación manual de cada paso.

SÍNTESIS FINAL

Lo esencial

Un sistema de subagentes no es simplemente «varios agentes de IA juntos»: es una arquitectura concreta —un orquestador que conserva el contexto y convoca trabajadores especializados, aislados entre sí, que devuelven resultados resumidos— diseñada para dividir tareas complejas en piezas manejables, paralelizar lo que se puede paralelizar y acotar los permisos de cada pieza. Ese mismo diseño es lo que permite tomar un proceso que hoy vive de forma implícita en la cabeza de una persona y convertirlo en un sistema explícito, repetible y, cuando corresponde, automático.

La decisión de sumar subagentes a un problema no debería ser automática: la evidencia de 2026 muestra con la misma claridad los casos donde aportan una ventaja real y los casos donde un agente único, mejor diseñado, sigue siendo la opción más simple, más barata y más confiable. Diseñar bien un sistema de subagentes empieza por saber distinguir entre esos dos escenarios antes de multiplicar la cantidad de agentes involucrados.