SOUTH SUMMIT · IE UNIVERSITY

Beyond the buzzwords: AI agents and the modern AI stack

Daniel Sierra
AI Engineer · IE Adjunct Faculty

self-driving
reasoning
MCP
GenAI
AI-powered
context window
inference
few-shot
LangGraph
multimodal
orchestration
agentic
embeddings
tokens
Agent
grounding
copilot
hallucination
fine-tuning
LangChain
workflows
chain of thought
AGI
generative AI
AI-native
memory
transformer
prompt injection
alignment
zero-shot
LLM
RAG
observability
tool calling
evals
guardrails
RLHF
autonomous
foundation model
prompt engineering
vector DB

LLM

Large Language Model

Just an autocompleter.

Given any text, it picks the most likely next word. Again, and again.

PROMPT

"The Spanish real estate market in 2026 looks ___"

NEXT-TOKEN CANDIDATES

increasingly 38%	hot 22%	uncertain 14%	strong 9%	…

1 · TEXT IN

"The Spanish real estate market in 2026 looks ___"

TOKENIZE

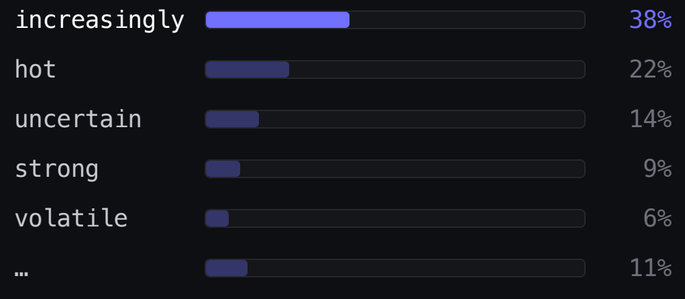
The | Span | ish | real | estate | market | in | 2026 | looks

9 tokens → vectors ∈ ℝⁿ

2 · THE MODEL

3 · PROBABILITY OVER VOCAB

TOP NEXT-TOKEN CANDIDATES



sample one token. append. run it all again. repeat.

How the model learned in the first place

1 · THE DATA

- every book ever scanned
- the public web
- all of GitHub
- Wikipedia, papers, news
- billions of images with captions
- hours of audio and video
- screen recordings, transcripts
- ...

everything, turned into tokens

→

2 · EMPTY NETWORK

random weights

8 billion random numbers.
the network knows nothing yet.

→

3 · THE LOOP

1 SHOW A SENTENCE
"Madrid is the capital of ___"

2 PREDICT THE NEXT TOKEN
Sydney 11%
Spain 28%
France 8%

3 CHECK AGAINST TRUTH
correct answer was "Spain"

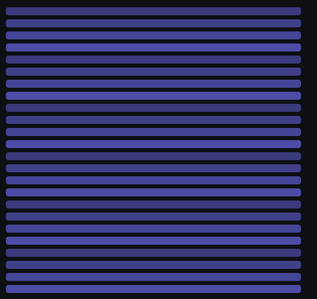
4 NUDGE THE WEIGHTS
so "Spain" scores higher next time

↻ repeat trillions of times

→

4 · BASE MODEL

tuned weights

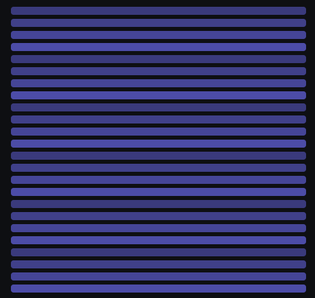


the same numbers,
now shaped to predict text.

From a base model to the one you actually chat with

1 · BASE MODEL

the trained model



knows a lot of text.
predicts the next token.
doesn't follow instructions.

→

2 · INSTRUCTION TUNING

SHOW IT IDEAL ANSWERS

Q: Explain photosynthesis.
A: Plants turn sunlight into energy by...

Q: Translate "thank you" to Spanish.
A: gracias

Q: Summarize this email in 1 line.
A: Q3 numbers ready, review by Friday.

... thousands more pairs

it learns the Q → A shape

→

3 · PREFERENCE LEARNING

MODEL WRITES TWO ANSWERS

Q: "Why is the sky blue?"

✗ Because of physics.

✓ Sunlight scatters off air molecules; blue scatters most, so we see blue.

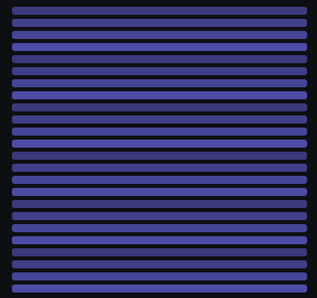
↓ humans rank. model learns the preference.

RLHF · DPO · ORPO

→

4 · INSTRUCT MODEL

ready to chat



ChatGPT, Claude, Gemini.
follows instructions.
refuses harmful requests.

CLOSED-WEIGHT

proprietary · API only

OpenAI

- GPT-5
- GPT-5.5
- o3

Anthropic

- Opus 4.8
- Sonnet 4.6
- Haiku 4.5

Google

- Gemini 3.5 Flash
- Gemini Omni

xAI

- Grok 4.3
- Grok Build

Alibaba

- Qwen 3.7 Max

OPEN-WEIGHT

downloadable · run anywhere

Meta

- Llama 4 Scout · 109B
- Llama 4 Maverick · 400B

Mistral AI

- Small 4 · 119B
- Medium 3.5 · 128B
- Large 3 · 675B

Google

- Gemma 4 · 2B
- Gemma 4 · 4B
- Gemma 4 · 26B
- Gemma 4 · 31B

Alibaba

- Qwen 3.6 · 7B
- Qwen 3.6 · 32B
- Qwen 3.6 · 72B

DeepSeek

- V4 · 671B
- R1 · 671B

Microsoft

- Phi-4 · 14B
- Phi-4 reasoning · 15B

01 · LLM · THE WORDS YOU KEEP HEARING

token	A piece of text. A whole word, or part of one. The model reads in tokens, not letters.	"real" → 1 token "Spanish" → "Span" + "ish"
vocabulary	The full list of tokens a model knows.	Llama 3 → ~128,000 tokens covers most words & word-pieces
embedding	Turning each token into a list of numbers, so the model can do math on language. Similar words get similar numbers.	"real" → [0.21, −0.45, ..., 0.87] ~4,000 numbers per token
transformer	The architecture. A deep stack of layers that refines those numbers until the model can guess the next token.	the T in GPT. Claude, Gemini, Llama — all transformers.
weights	The numbers the model learned during training. More weights = more it can store.	Llama 3 8B → 8 billion numbers GPT-4 → trillions (closed)
context window	How much text the model can hold in mind at once. The prompt plus the conversation so far. Older stuff falls off the edge.	Claude → 200,000 tokens (~150k words) Llama 3 → 8k–128k
hallucination	When the model invents something that sounds right but isn't. It's still autocompleting, with no idea whether it's true.	"Who won the 2026 Champions League?" → a confident, plausible, wrong answer.
fine-tuning	Taking a pre-trained model and training it a bit more on a smaller, focused dataset to shape its behavior.	start with a generic LLM → feed it 10k proprietary Q&A pairs → it learns your voice and vocab.

Brilliant, but boxed in.

FROZEN IN TIME

Knowledge stops at the training cutoff. Doesn't know what happened yesterday.

ISLANDED

Can't see your files, your APIs, your business data.

GENERALIST

Wide but shallow. No expertise on your specific problem.

AMNESIAC

Forgets every message the moment it's answered. Each call starts from zero.

↓ TO CLOSE THE GAPS, GIVE THE MODEL TWO THINGS ↓

Tools

Let the model reach into the real world: fetch fresh data, query your systems, take actions.

- → web search · the live internet
- → terminal & code · shell, Python, scripts
- → function calls · invoke specific actions
- → external services · via **MCP** or CLI
- → proprietary documents · via **RAG**

Memory

Let the model carry state across turns, across sessions, across days.

- → short-term · this conversation
- → long-term · between different conversations

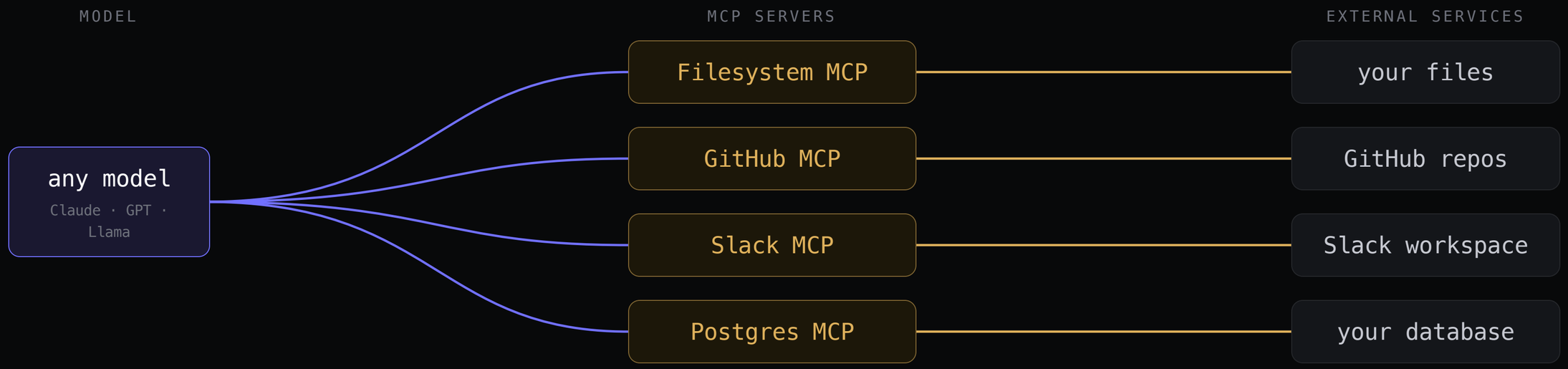
CONCEPT 02 / 04

MCP

Model Context Protocol

USB-C, but for AI tools.

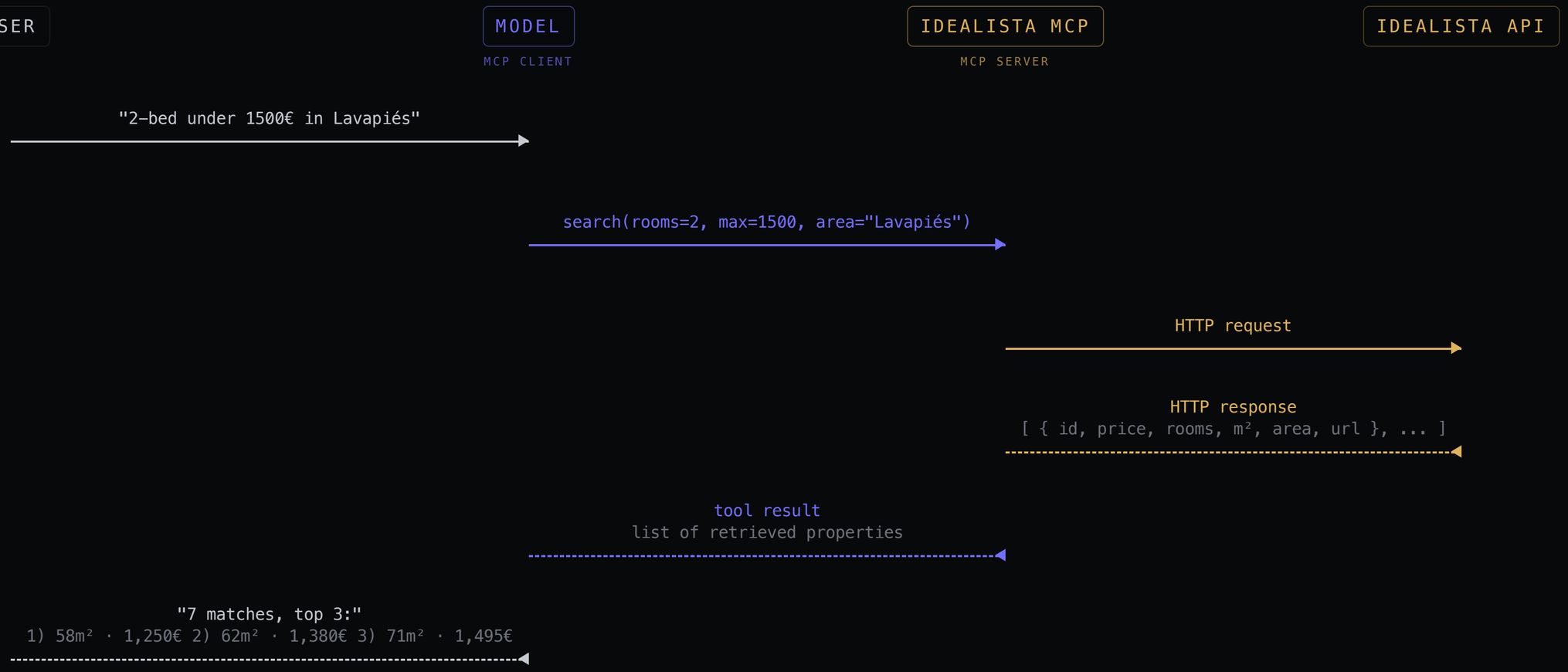
One protocol that lets any model talk to any service. Build the integration once, every MCP-capable app can use it.



the model speaks one protocol → each MCP server wraps one external service.

Idealista MCP

One question, traced through the lanes.



CONCEPT 03 / 04

RAG

Retrieval-Augmented Generation

Grounded, not guessing.

Before answering, the model fetches the most relevant pieces of your data and reads them in.

WITHOUT RAG

question → model → best guess from training data

stale knowledge. no access to your docs. confident wrong answers.

WITH RAG

question → search your docs → top chunks → **model** → grounded answer

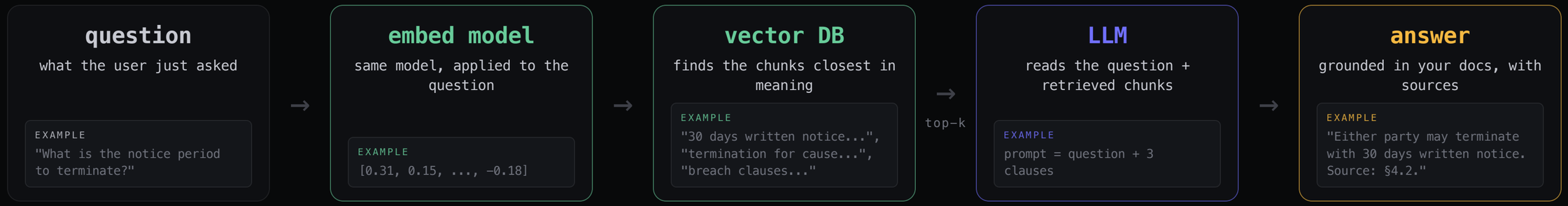
fresh data, your data, with sources you can cite.

index once. the model reads only the relevant pages, every question.

The RAG pipeline, two phases

01 Indexing done once, ahead of time

02 Querying runs on every user question



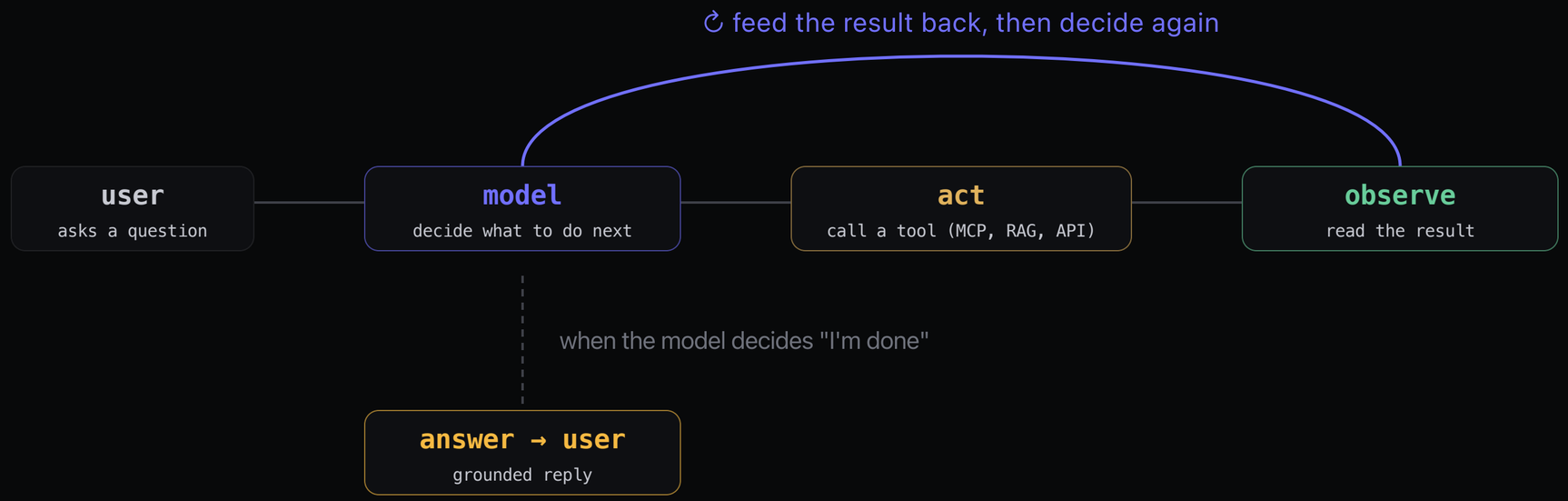
CONCEPT 04 / 04

AGENT

Autonomous Agent

A model that drives itself.

The model decides what to do next, every turn, on its own.

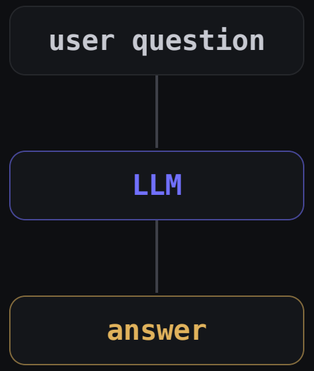


this topology has a name: **ReAct** · Reason + Act · the dominant agent pattern today

Workflow, or agent?

The difference between a workflow and an agent isn't the tools. It's who decides the path.

WORKFLOW



The developer writes the arrows, at build time.

AGENT

The model writes the arrows, at runtime, every turn.

the developer wires the components. the model wires the path.

Same ticket. Two paths.

ticket → "I haven't received my order #1234. It's been 5 days."

WORKFLOW

fixed triage path

LLM
classify the ticket type

↓

Open Issue
open a ticket with the issue

↓

Reply
send templated reply to the customer

↓

⚠ human agent
actually resolves the issue

↳ reply: "We are checking on order #1234. We will reply within 24h."

AGENT

contextual handling

LLM
reasons, picks the right tool, decides when to stop

⇄

TOOLS

orders db look up an order

⇄

customers db look up a customer

Thought: Check the order status first.

Action: Check order 1234 (Tool call)

Observation: Shipped Monday to address A.

Thought: Is A still the customer's address?

Action: Look up the customer's current address (Tool call)

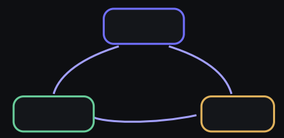
Observation: Customer changed address on Tuesday — now address B ⚠

↳ reply: "Order #1234 was shipped to your previous address. We can reroute to your updated one — please confirm."

ReAct is the common one. Not the only one.

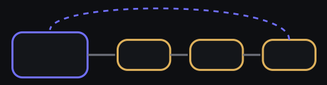
Other agent shapes. In every one the model still owns the loop. Only how the work gets organized changes.

ReAct



Think, act, observe, repeat. The default loop.

plan-and-execute



Plan all the steps first, then run them. Replan if needed.

reflexion

Act, grade its own output, retry smarter.

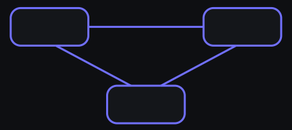
tree-of-thoughts

Explore several branches, keep the best one.

orchestrator-workers

A lead splits the task and delegates to sub-agents.

multi-agent



Specialist agents hand off to each other.

every one is an agent: the model owns the loop and chooses the path. · real systems often nest a few of these

ACT 2

Let's build one, live, layer by layer.

Daniel Sierra

AI Engineer · IE Adjunct Faculty

EMAIL dasirra@gmail.com

LINKEDIN linkedin.com/in/dasirra

WEB danisierra.dev

X @dasirra1