ETH zürich

IVIA lab

AI in Practice & Agentic AI

From the Right Tool to Autonomous Workflows

Matthias Miller

emba X5 - Essentials III: Cognitive Technologies

July 8, 2026

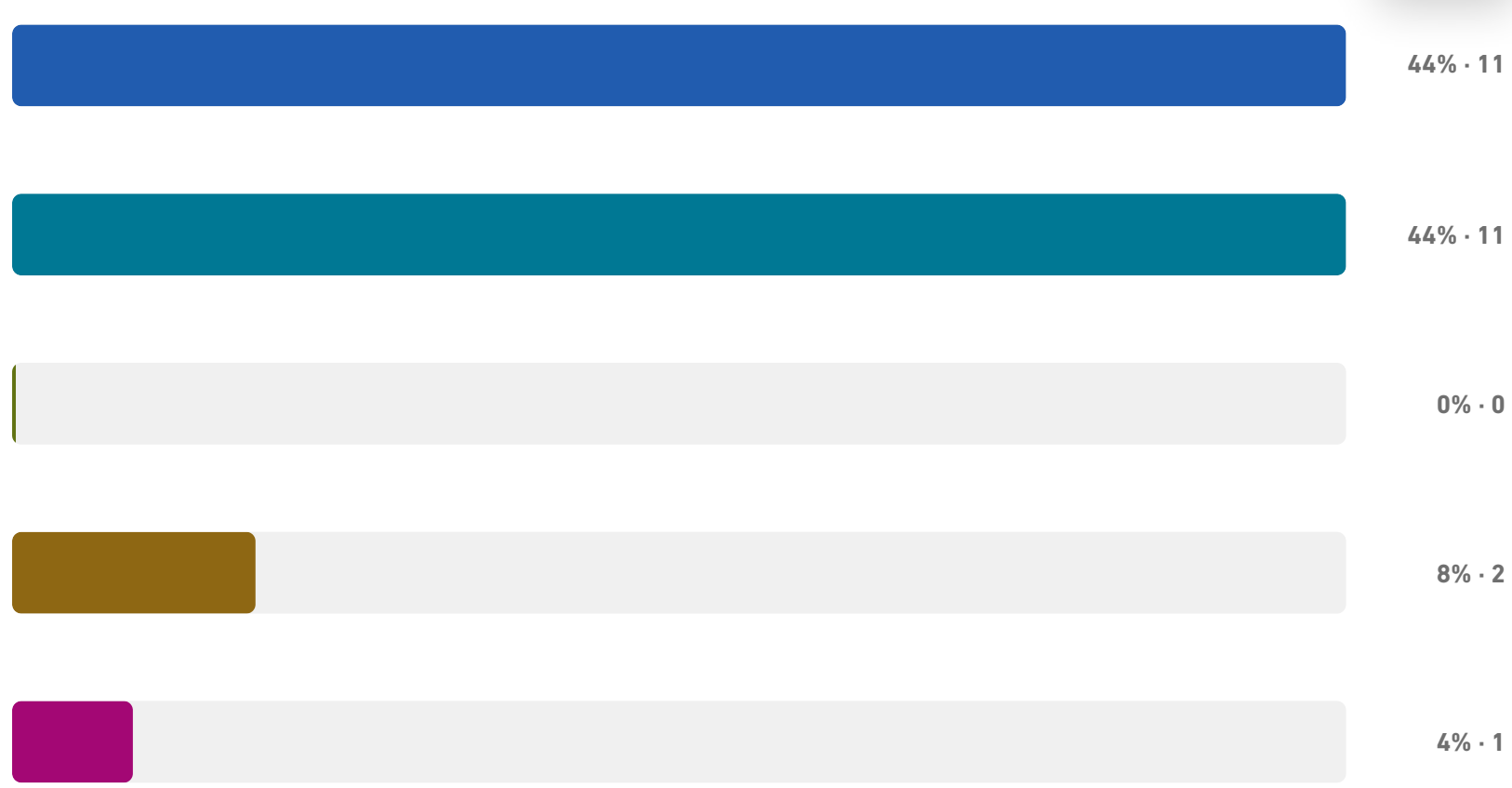
Task Overview

Structured (Tables / Numbers / Dates)	Semi-Structured (Text + Audio)	Unstructured (Multimodal)	Digital Transformation
Classification	Tagging	Research	Template Creation
Reporting	Text Editing	Versioning	Online Forms
Reconciliation	Spell-Checking	Fact-Checking	Process Optimization
Scheduling	Transcription	Revision	Process Automation
Data Processing	Text Generation	Information Extraction	Process Transformation
Planning	Translation	Ideation	Document Structuring
	Summarization	Content Creation	Data Formatting
			Programming

Which recurring task costs you the most time?

MULTIPLE CHOICE

Which data type do you work with most?



Guiding Questions for Choosing an AI Tool

1. Data types and requirements

Which data types (text, etc.) must be processed?

2. Data privacy, usage, legal & ethical compliance

Which data-protection requirements and ethical standards must be met?

3. Cost model

Which cost model (free, hybrid/freemium, subscription or one-time payment) fits the available budget?

4. Onboarding effort (usability)

How much time and resources are available for onboarding/training?

5. Transparency and explainability

How important is it that results are traceable/explainable?

6. Integration into existing systems (APIs)

Does the tool need to integrate into existing workflows?

7. Scalability

Can the tool handle increasing data volumes?

8. Adaptability and flexibility

Is there a need for custom adjustments/configurations?

9. Compatibility and platform dependency

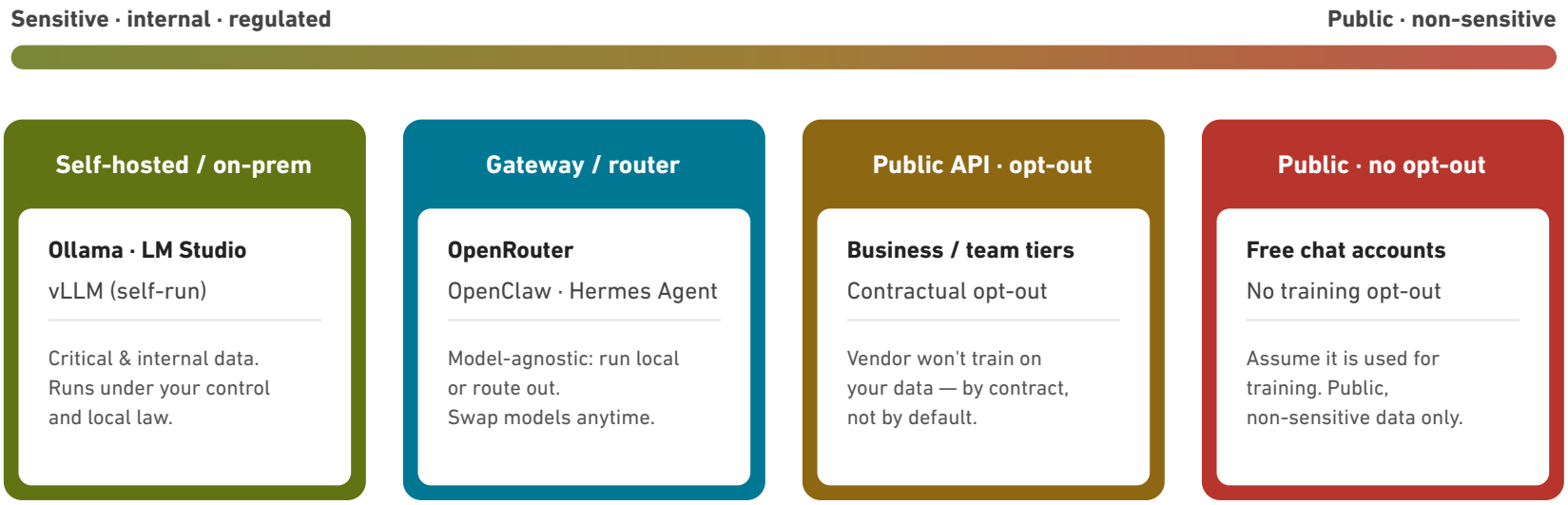
On which platforms (OS, mobile) must the tool be used?

10. Programming skills

Are programming skills required for effective use?

Where does the model run?

Match hosting to how sensitive the data is



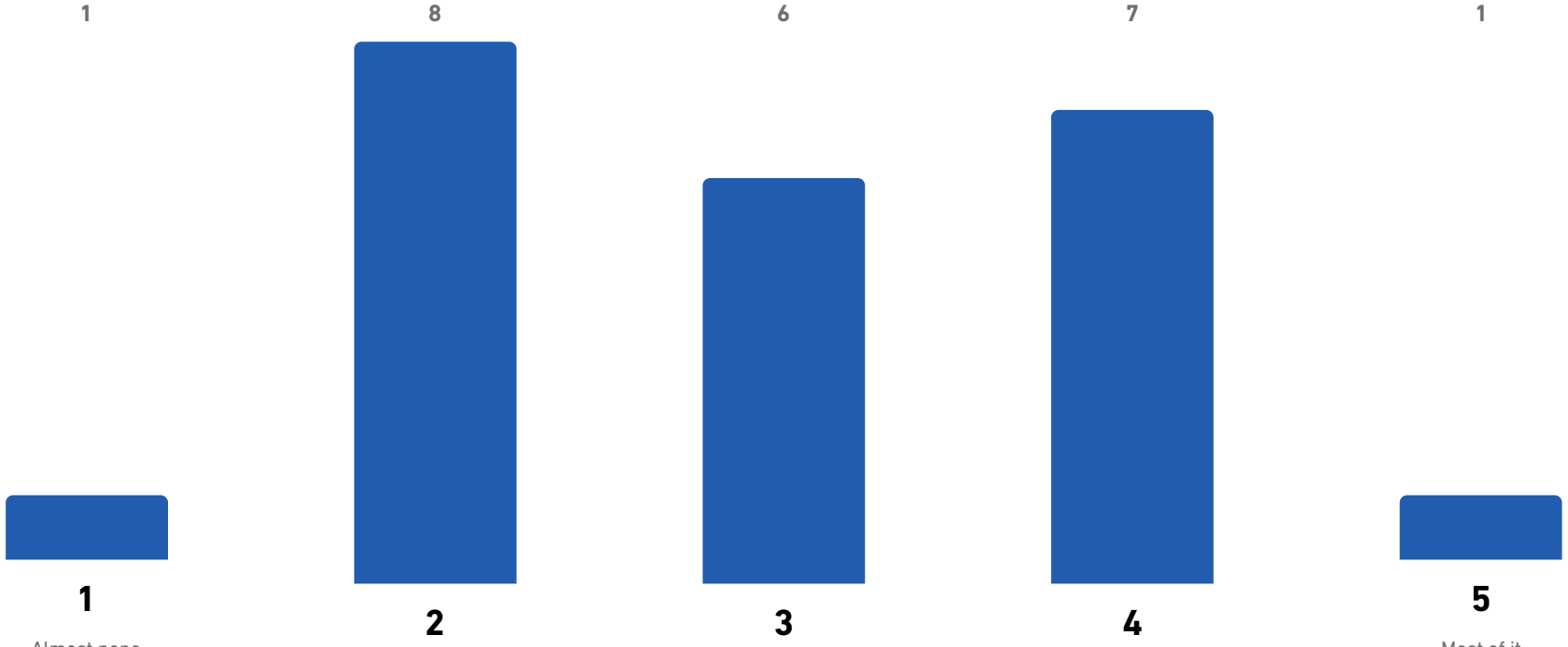
The more sensitive the data, the further left it must stay.

Same task, different tier — pick the leftmost tier the data allows.

RATING SCALE

How much of your workday could today's AI tools sensibly support?

3.0 average · 23 responses



From Tasks to Skills

Turning recurring work into reusable, composable building blocks

Start from the Process

Look at what you already do — repeatedly

- Take a **real, recurring process** — e.g. handling an incident report, onboarding a supplier, triaging emails
- Break it into its **individual steps**
- Notice which steps **recur** — within this process and across others

The goal is not to automate the *whole* process at once, but to find the **repeating building blocks** underneath it.

What is a Skill?

Reusable know-how, packaged so an agent can apply it reliably

Skill

A packaged workflow for doing a **specific task well** with your rules, examples, templates, and checks.

MCP

A standard interface that lets models connect to tools, data sources, and reusable skills.

Curate and verify first. Automate only once the skill is proven, then scale it when it is safe to make public.

What makes a Skill effective: components, ranked

04 · ANATOMY OF A SKILL

1	**Description / Trigger**	decides whether anything else runs	ALWAYS ON
2	**Body (the procedure)**	the judgment a base model wouldn't infer	ON TRIGGER
3	**Scripts**	offload deterministic, fragile work	ON DEMAND
4	**Assets**	exact fidelity: fonts, logos, templates	ON DEMAND
5	**References**	on-demand docs for large / variant knowledge	ON DEMAND
6	**Examples**	calibrate a specific output format	ON DEMAND
7	**Env vars / API access**	integration boundary to third-party systems	RUNTIME

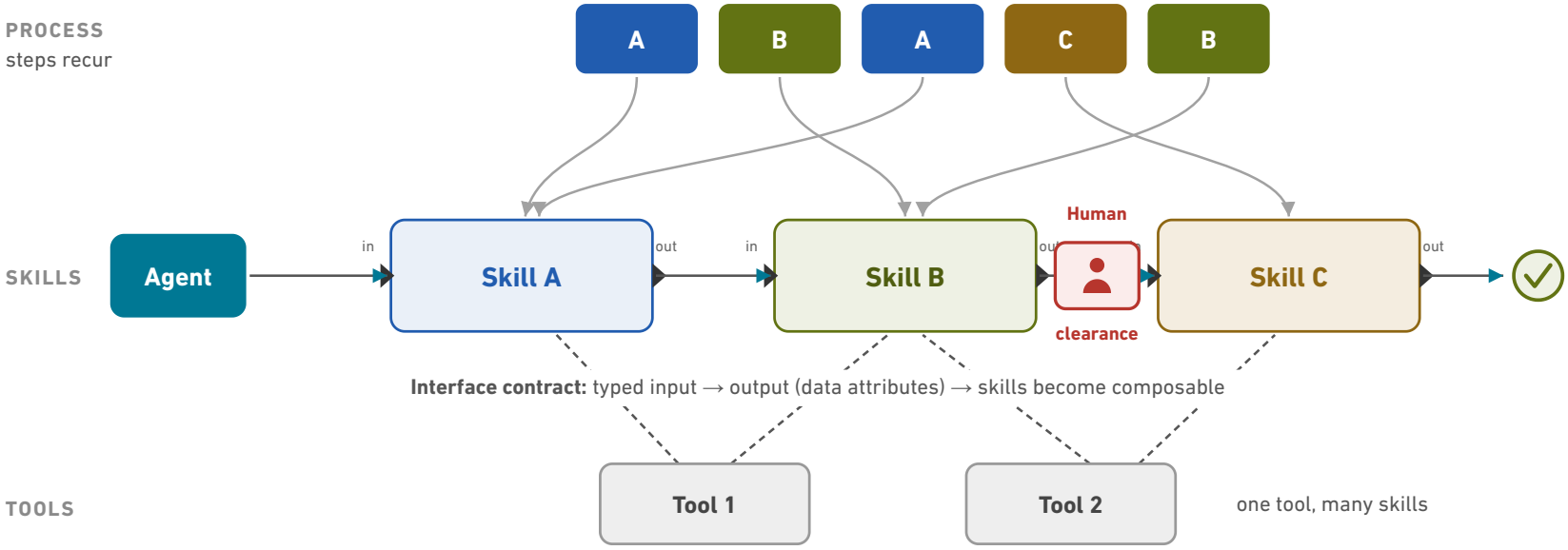
The tier tags mirror progressive disclosure: keep the core always visible, load detail only when needed.

Extract Skills

Each recurring step becomes a reusable **skill** — with a contract, and maybe tools

- A **skill** captures *how to do one step well* — defined once, reused everywhere
- Give it a clean **interface contract**: typed **inputs** and **outputs** — which **data attributes** go in, which come out
- **Tool calling** can be *part* of a skill — the skill decides *when* and *how* to use a tool
- Both a skill and a tool are a **service** behind a contract — so one **tool** serves **many skills**
- **With clear contracts, skills become composable — arrange them in any sequence to build up a workflow**

From Process to Orchestrated Skills



The **agent** sequences skills — passing each output to the next input — to complete a larger process.

Tools vs Skills

The difference — and why it matters

TOOLS = capabilities

what an agent CAN do (verbs)

- Search the web
- Run code
- Query a database
- Send an e-mail

SKILLS = know-how

HOW to do OUR task well

- Write our incident report
- Follow our brand voice
- Process our file format

+ our context: template, rules, examples

a skill decides WHEN & HOW to call tools

Tools are the verbs. Skills are the recipes that use them for your job.

Open standards like MCP connect an agent to tools, data, and skills.

MULTIPLE CHOICE

"Write our incident report" — is that a tool or a skill?

Tool Calling

What happens if you ask an LLM for the current time?

What time is it right now?

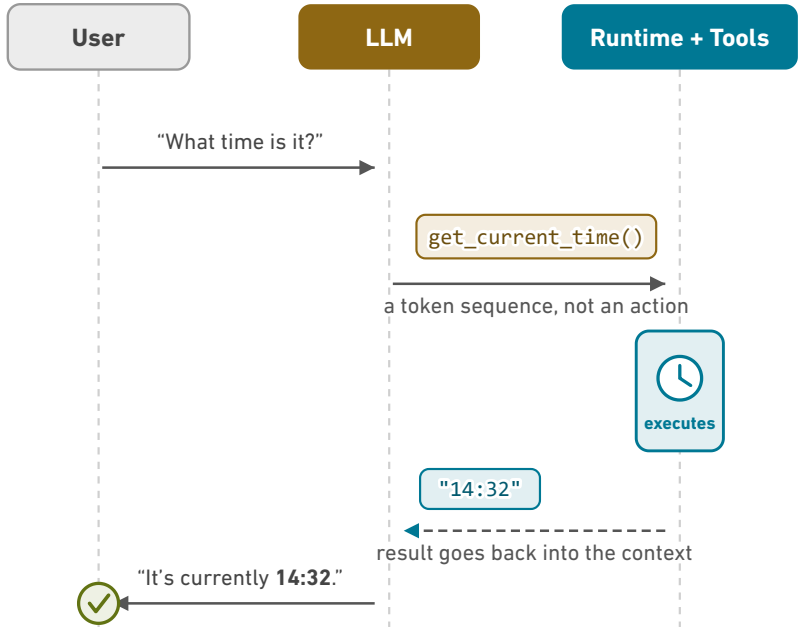
It is currently **14:32**.

- Confidently stated, perfectly formatted, and plausible –
- **... but almost certainly wrong**

Tool Calling

Giving the Prediction Machine a Way to Act

- We describe a set of **tools** (functions) the model is allowed to use
- When it needs something it **cannot know**, it stops guessing and emits a **structured request**, e.g. `get_current_time()`
- The returned information is fed back into the context, and the model continues with the new information

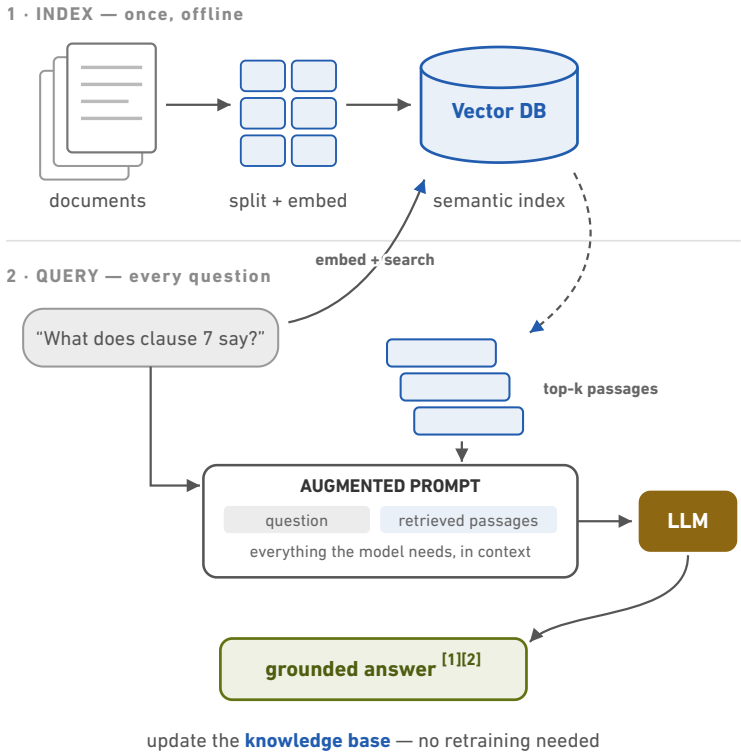


The **LLM** never executes anything — it only predicts tokens.
The **runtime** runs the call and feeds the result back.

Retrieval-Augmented Generation (RAG)

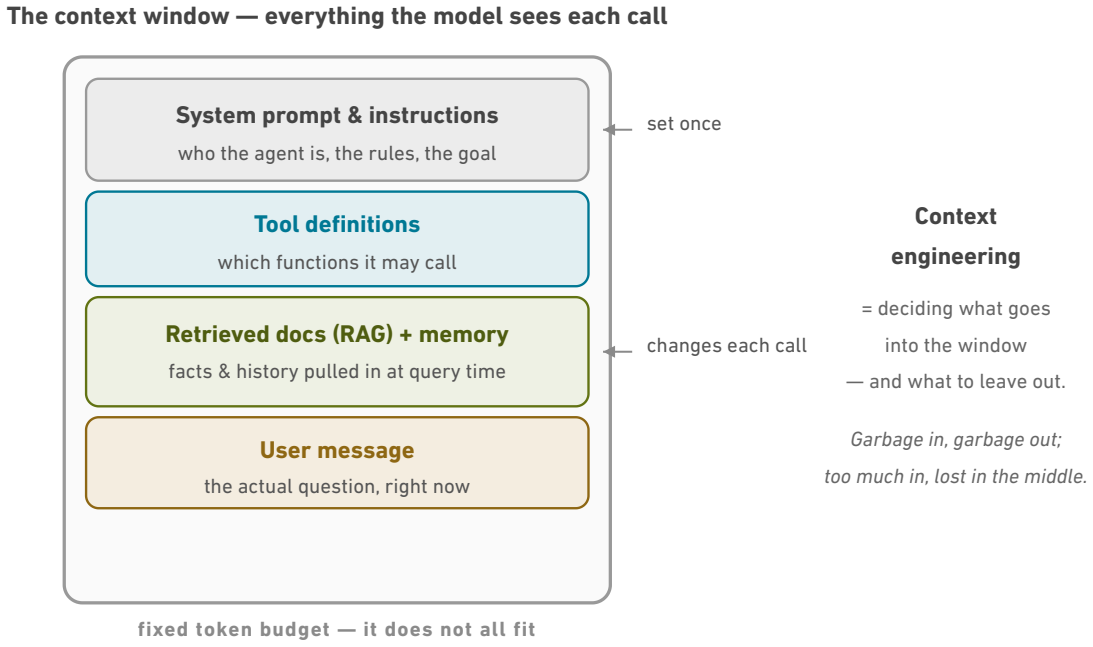
Grounding the Model in External Knowledge

- An LLM only knows its **training data** and has no information about the world after that point
- **RAG** retrieves relevant passages from a **knowledge base** at query time
- The retrieved text is added to the **context**, so the answer is **grounded** in real sources



Context Engineering

What you put in the window is what you get out



Context engineering

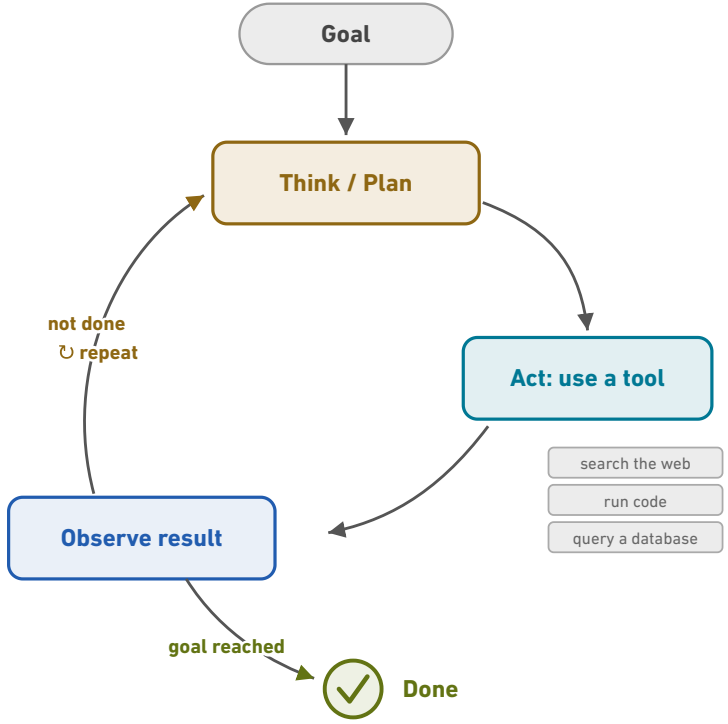
= deciding what goes into the window — and what to leave out.

Garbage in, garbage out; too much in, lost in the middle.

Agents

LLM + Tools + a Loop

- An **agent** places an LLM in a **loop** with tools and a **goal**
- Each cycle: **think** → **act** (call a tool) → **observe** the result → repeat
- Capabilities grow with context: the more tools and information, the more complex the tasks it can solve

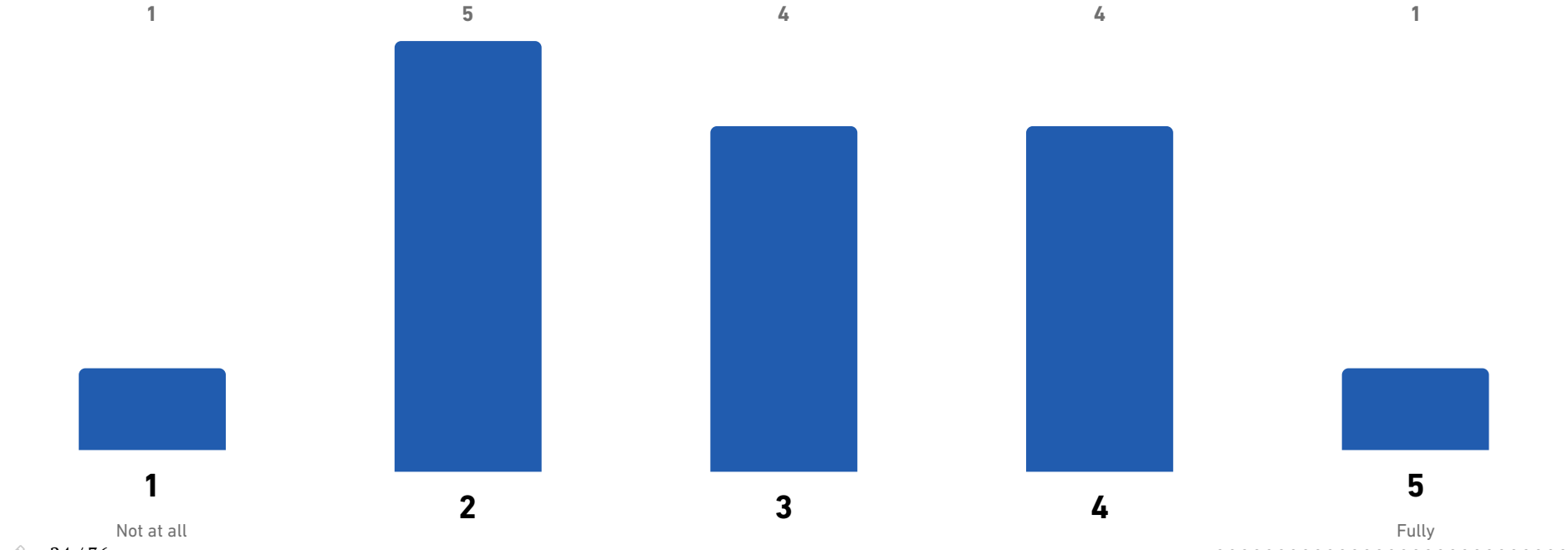


Each cycle, the **LLM** sees everything so far — and decides the next step itself.

RATING SCALE

How much would you trust an AI agent to act autonomously on your behalf?

2.9 average · 15 responses

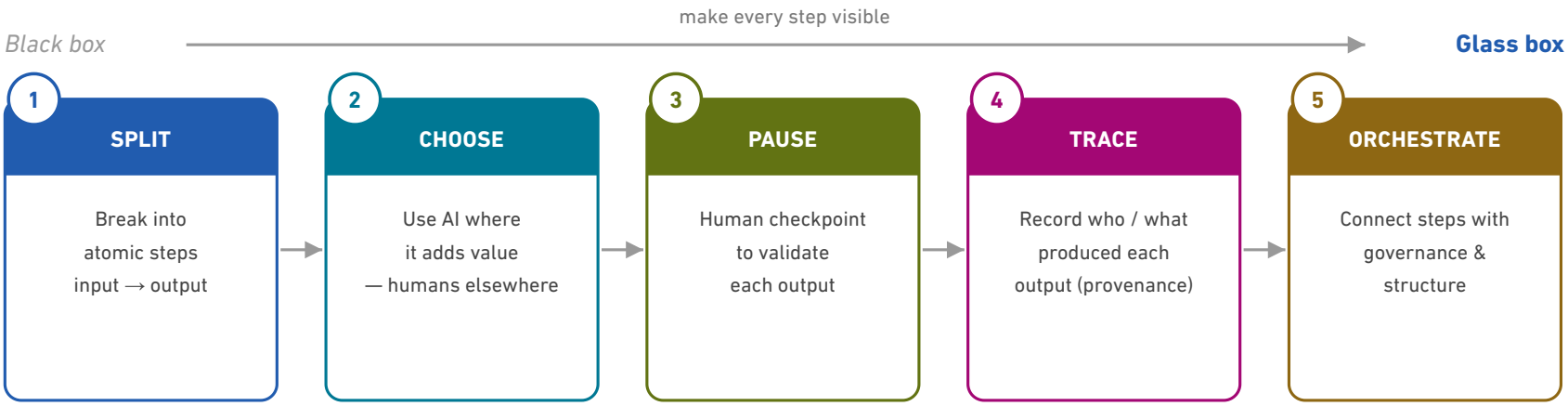


A tool solves a task.

Your day is made of **processes** — let's make them transparent

From Black Box to Glass Box

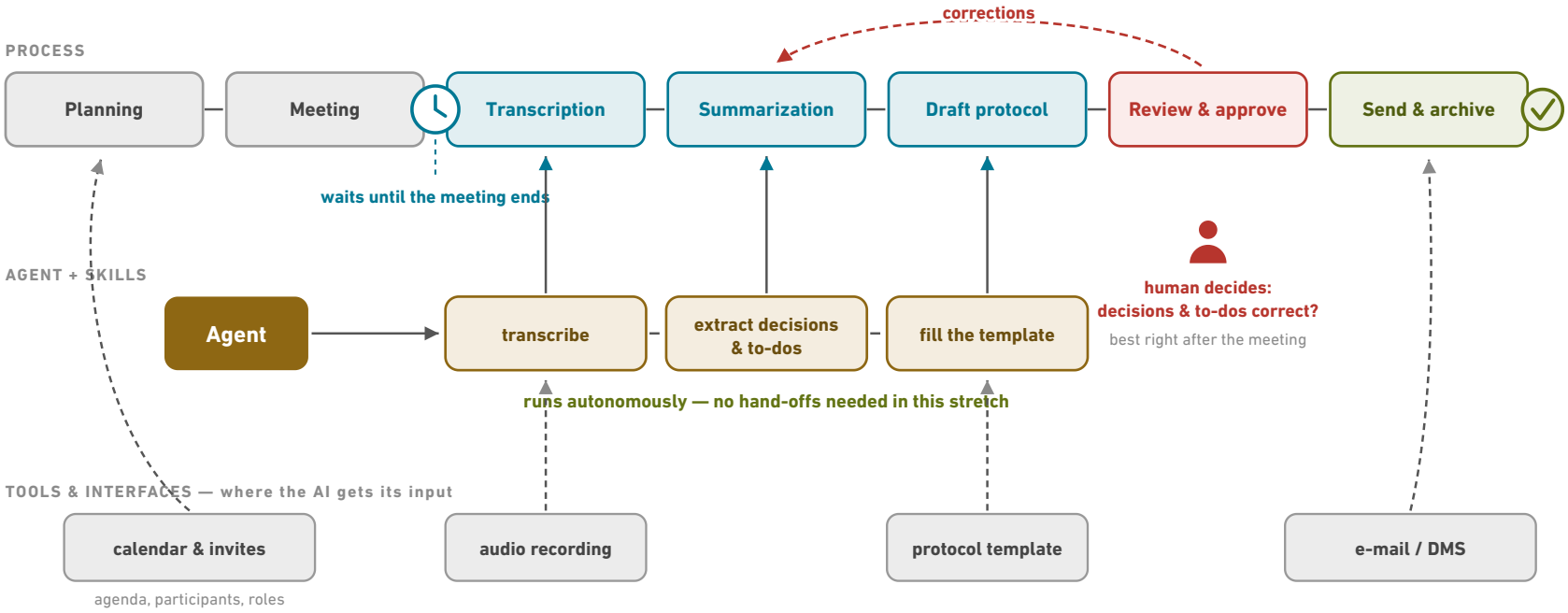
Five moves to make a process transparent and governable



Five moves turn an opaque workflow into a transparent, governable one.

AI drafts; humans decide. Every output stays attributable.

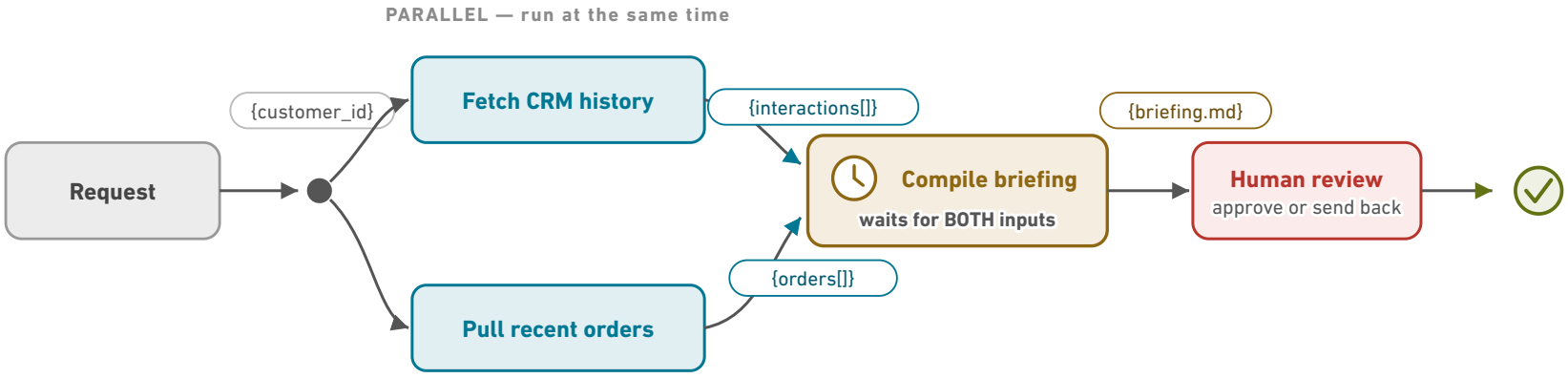
Putting It Together: From Meeting to Protocol



Autonomous where the work is **mechanical** — a gate wherever **judgment and accountability** live.

Pipelines Aren't Straight Lines

Split, merge, and wait — held together by input/output contracts



Not a straight line: branches run in parallel, a step waits until every input arrives.

Typed input / output contracts on each edge keep the pieces composable and checkable.

Security, Safety & Control

More Autonomy, More Ways to Go Wrong

When Autonomy Backfires

Agents Inherit Every Weakness of the LLM – and Add New Ones

Prompt injection

Malicious instructions hidden in a web page, email, or document hijack the agent's next steps.

Excessive permissions

Broad tool access turns a single mistake into deleted data, leaked secrets, or unwanted actions.

Compounding errors

In a loop, a small misstep snowballs across many autonomous actions (cf. AutoGPT).

Accountability

When an agent acts on its own, who is responsible for the outcome?

Designing for Safety

Keeping Agents Useful *and* Trustworthy

Least privilege

Give each tool the narrowest scope it needs; sandbox code execution and file access.

Human in the loop

Require explicit approval before high-impact or irreversible actions.

Guardrails

Provide soft or hard constraints on the agents actions. Isolate untrusted content from trusted instructions.

Observability

Log every step so actions can be audited, traced, and rolled back.

So many candidates — pick one and SPLIT it

Everyday processes that reward the Glass-Box treatment

- **Document translation**
- **Recruitment screening**
- **Data cleaning & quality checks**
- **Knowledge retrieval & research**
- **Customer-feedback analysis**
- **Conference registration**
- **Training-material creation**
- **Email triage & drafting**
- **Fraud / anomaly detection**
- **Meeting protocols**

Reflection

Let’s discuss — no right answers

- Which of your processes is really a **black box** today — even to you?
- Where would you **never** let AI decide — and what must an agent **never do without you**?
- What data of yours could **never leave the building**?
- **Who is responsible when an autonomous agent gets it wrong?**

Take-Home Messages

AI in Practice & Agentic AI

- Start from your **data types and task types** — they narrow which tool fits; run the **guiding questions** (privacy, cost, integration, …) before adopting
- **Match the model to the data**: self-hosted for sensitive data, public only for the public — and pick by category (general vs. specialised)
- Break recurring processes into **reusable skills** with typed **interface contracts** — one **tool** serves many skills
- An **agent** is an LLM in a **loop** with tools and a goal — it only predicts tokens; the **runtime** executes (tool calling, RAG)
- **Context engineering**: what you put in the window is what you get out
- Turn processes into **Glass Boxes**: SPLIT → CHOOSE → PAUSE → TRACE → ORCHESTRATE — AI drafts, **humans decide**, every output stays traceable
- **Autonomy adds new risks (prompt injection, over-permissioned tools, compounding errors) → least privilege, human oversight, auditing**

Thank You!

Questions?