NASA AI/ML STIG · 14 SEP 2026

Foundations of LLM agents, and the state of the art

How the models work, why they work, and what you build around them.

Josh Speagle
Astronomy & Astrophysics · Statistical Sciences, University of Toronto

David A. Dunlap Department of Astronomy & Astrophysics
UNIVERSITY OF TORONTO

JOSHSPEAGLE.COM

Created with assistance from Claude

Built with Claude. Every idea is mine. None of the slides are.

How this talk was built is the example in part 3.

Four things to take away

How they work

Context in, a distribution out, one draw appended. Tool results return text into the context.

Part 1 · 12 min

Why they work

Varied text, no memorizing, a hard objective: general strategies.

Part 2 · 8 min

The harness

Everything around the model. You build it and you control it.

Part 3 · 14 min

Working with them

Clear vision, clear communication, clear execution.

Part 4 · 8 min

Where we are

How they work

Context in, a distribution out, one draw appended. Tool results return text into the context.

Part 1 · 12 min

Why they work

Varied text, no memorizing, a hard objective: general strategies.

Part 2 · 8 min

The harness

Everything around the model. You build it and you control it.

Part 3 · 14 min

Working with them

Clear vision, clear communication, clear execution.

Part 4 · 8 min

Text

Text becomes tokens

SUB-WORD PIECES, AND THEIR INDICES

Ast	roph	ysics	_is	_a	_statistical	_science
62152	22761	17688	374	264	29564	8198

A token is a piece of text with a fixed index in a vocabulary of about 100,000. The model sees only the indices.

Tokenizer: OpenAI cl100k_base · try one at tiktokenizer.vercel.app

The context is all the model sees

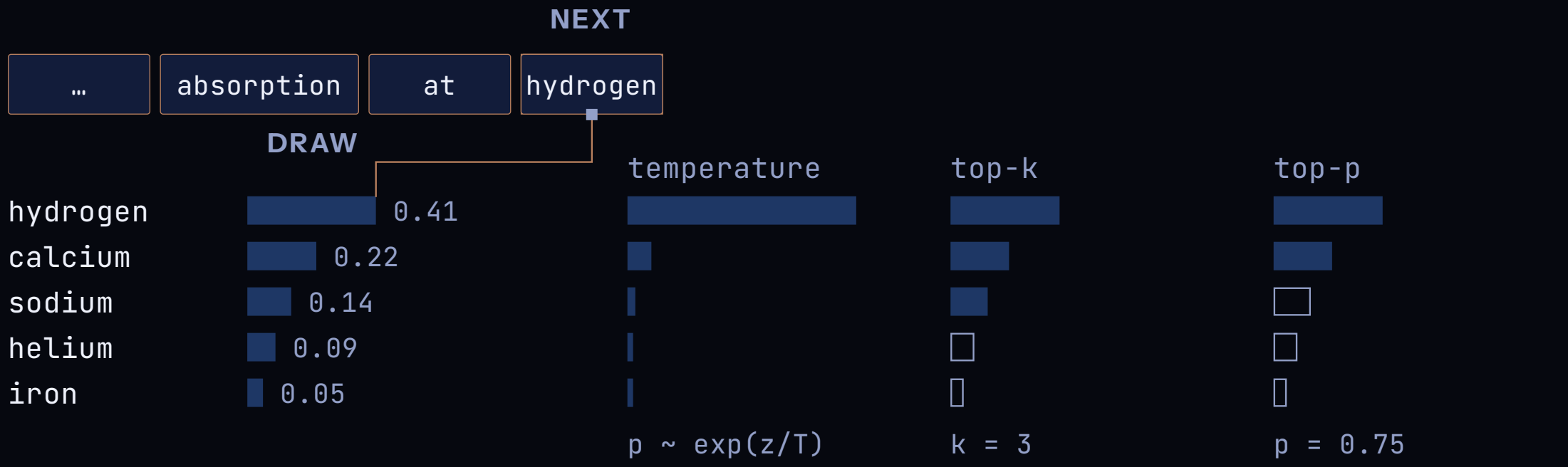
Everything the model knows about the problem is in this strip. Nothing carries over between runs unless it is put back in.

One pass, one distribution

One forward pass gives one probability for every token in the vocabulary. That is the whole output.

Values illustrative.

The next token is a random draw

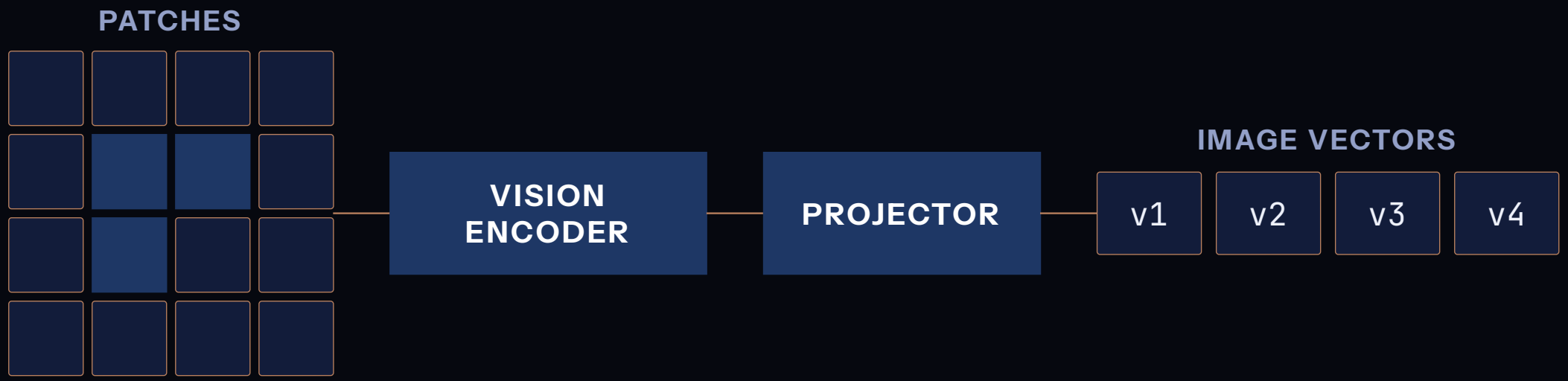


The draw is why one prompt gives different answers. Temperature, top-k and top-p reshape the bars first; none of them checks anything.

Not repeatable even at T = 0: the answer depends on what else is in the server's batch (Thinking Machines 2025).

Images, audio, and other inputs

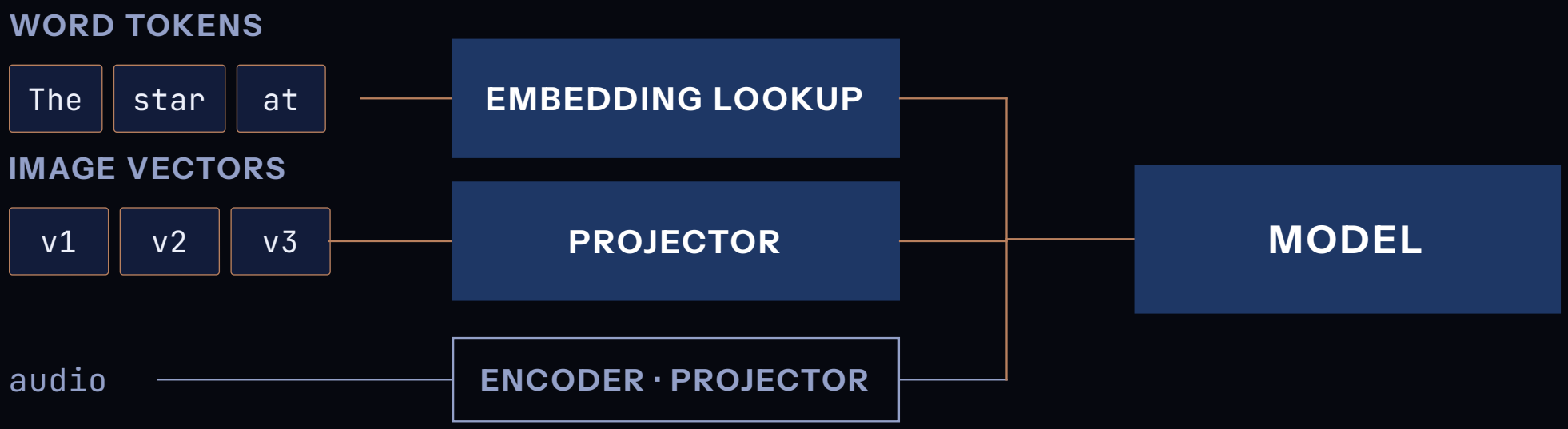
An image becomes vectors



Patches go through a vision encoder, then a learned projector into the model's embedding space. The result is continuous vectors, not vocabulary entries.

LLaVA-1.5: 576 per image · Qwen2-VL: 256–1024, set by resolution

One sequence, several kinds of token



Text tokens are looked up in a table; image and audio vectors arrive already embedded. From here on the model treats them all the same way.

The cost is real: an image is a few hundred tokens of context.

Tools

How a model refers to a tool

Described in the prompt

The harness lists each tool's name, description and arguments. The model writes `{"name": …, "input": {…}}`; the harness parses it.

As special tokens

Open models have vocabulary entries for it: `◁python_tag▷`, `<tool_call>`, `[TOOL_CALLS]`. End-of-turn and `<think>` are the same kind.

Both are learned in post-training. The tool itself is code the harness runs.

What a tool looks like

A TOOL DEFINITION

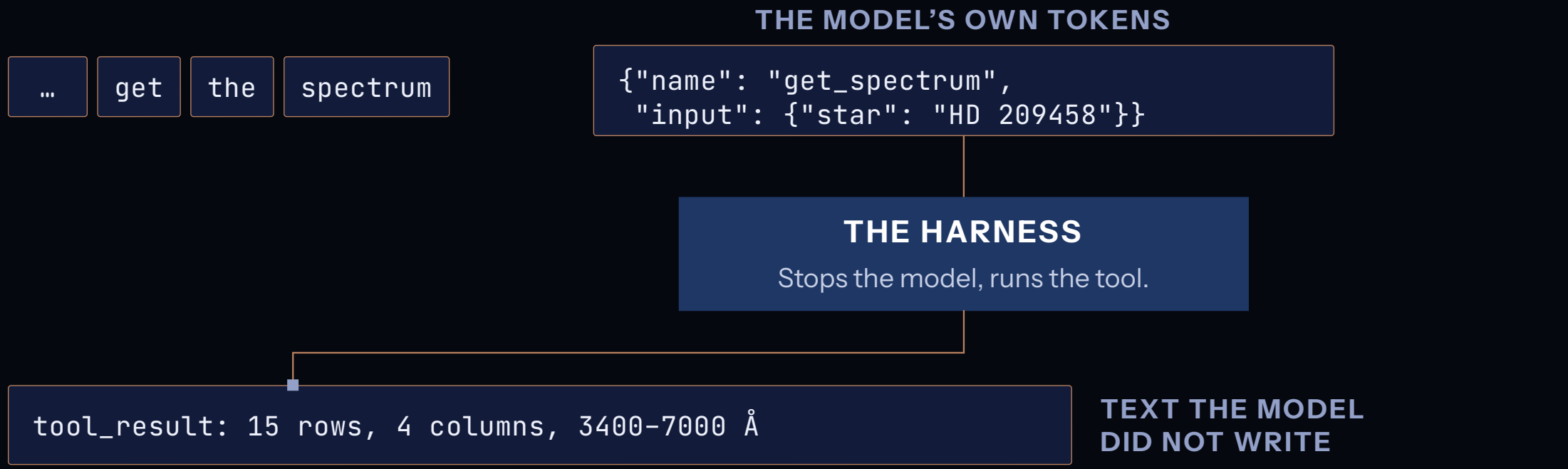
```
name: Read
description:
  Reads a file from the local filesystem
input:
  file_path  string, absolute
  offset     integer
  limit      integer
```

AND ANOTHER

```
name: Bash
description:
  Executes a bash command
input:
  command  string
  timeout  integer, ms
```

A tool is a function with a name, a description and typed arguments. That is all the model ever sees of it.

When the model calls a tool



The harness stops the model at the call, runs the code, and appends the result. The model continues from a context it did not fully write.

Prompt in, k tokens back

Until now every token in the context was chosen by the model or by you. A tool result is neither, and it can be any length.

The same call, three results

ONE TOOL CALL

```
{"name": "get_spectrum",
 "input": {…}}
```

THREE RUNS, THREE RETURNS

```
tool_result: 15 rows, 4 columns, 3400-7000 Å
```

```
tool_result: timeout
```

```
tool_result: 0 rows: no match
```

Results vary from call to call and can contradict what the model assumed. The model has to read them, and errors in them get acted on.

A model predicts the next token
from its context.
An agent is that model with tools,
and every result goes back into the context.

Where we are

How they work

Context in, a distribution out, one draw appended. Tool results return text into the context.

Part 1 · 12 min

Why they work

Varied text, no memorizing, a hard objective: general strategies.

Part 2 · 8 min

The harness

Everything around the model. You build it and you control it.

Part 3 · 14 min

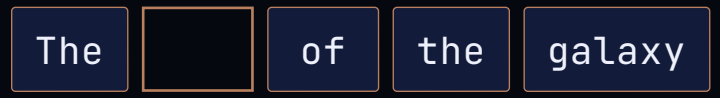
Working with them

Clear vision, clear communication, clear execution.

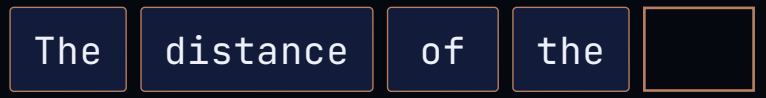
Part 4 · 8 min

The training task

MASKED · BERT, 2018



NEXT TOKEN · GPT-2, 2019



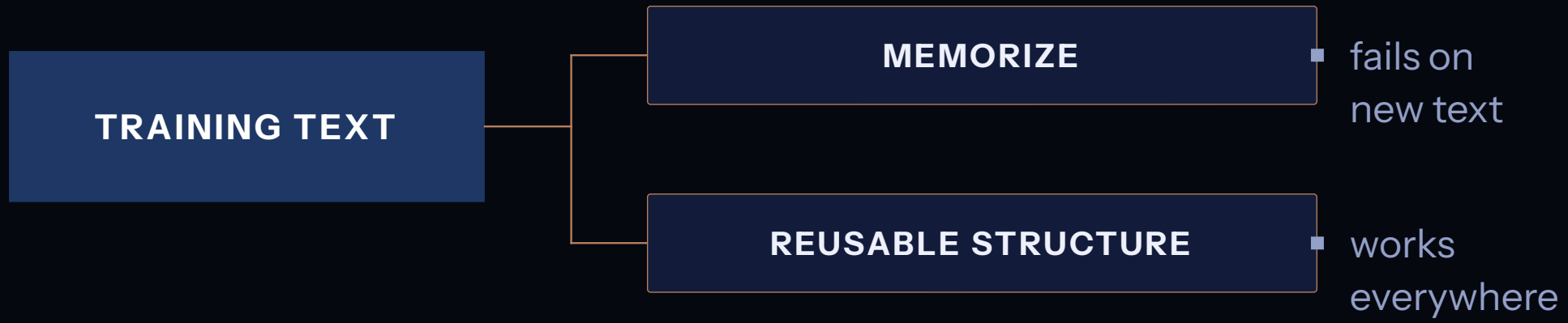
the same sentence,
continued left to right

REUSABLE STRUCTURE

Predict the hidden or next token, across trillions of tokens of text that is different every time.

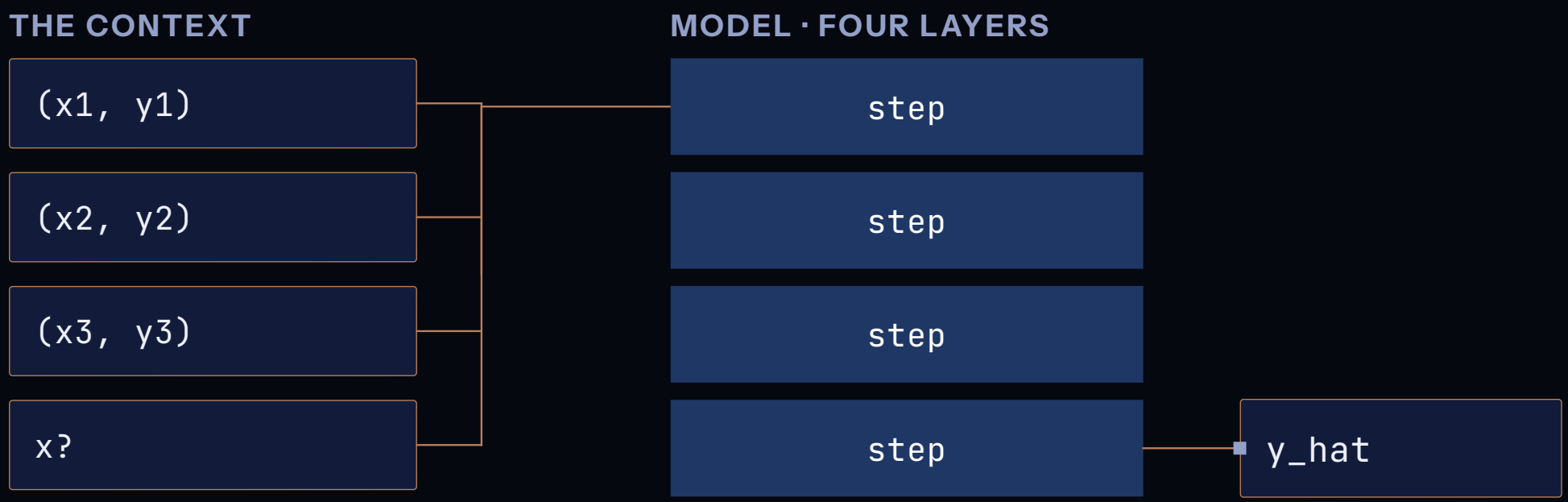
Masking (BERT 2018) and next-token (GPT-2 2019) both transferred to tasks never trained on.

Memorizing fails, strategies survive



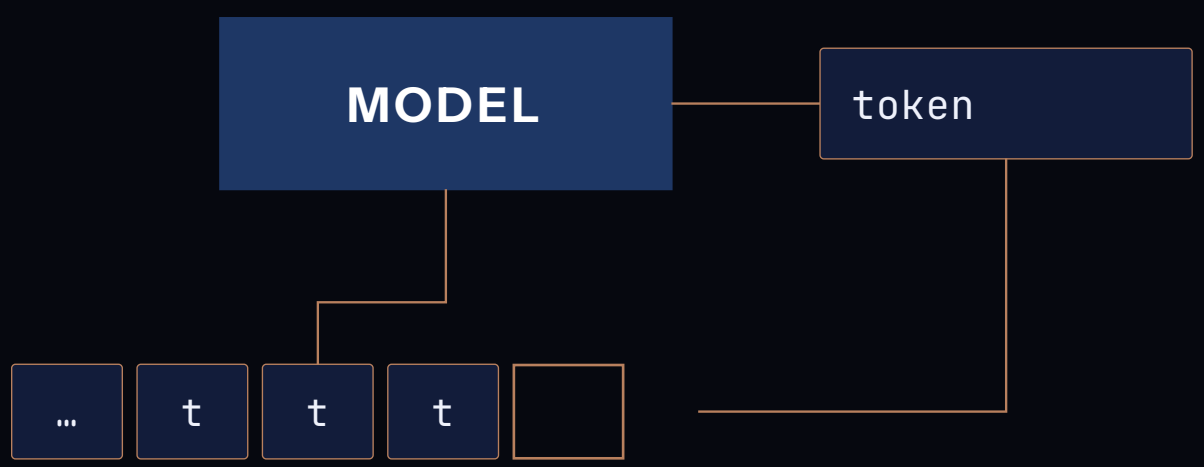
New text at every step, parts hidden, parts of the network dropped out. Whatever only works on seen examples is punished; whatever works everywhere is kept.

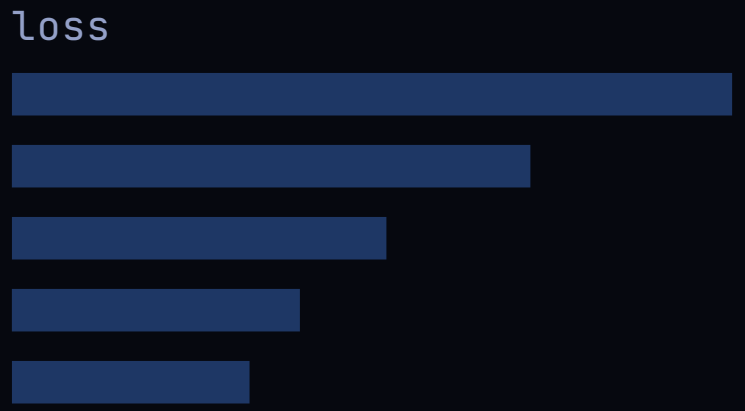
One layer, one gradient step



If the only strategy that works on every regression problem is gradient descent, the network learns gradient descent. One attention layer can implement one step (von Oswald 2022).

Generating tokens adds steps





THE CONTEXT, ONE TOKEN LONGER

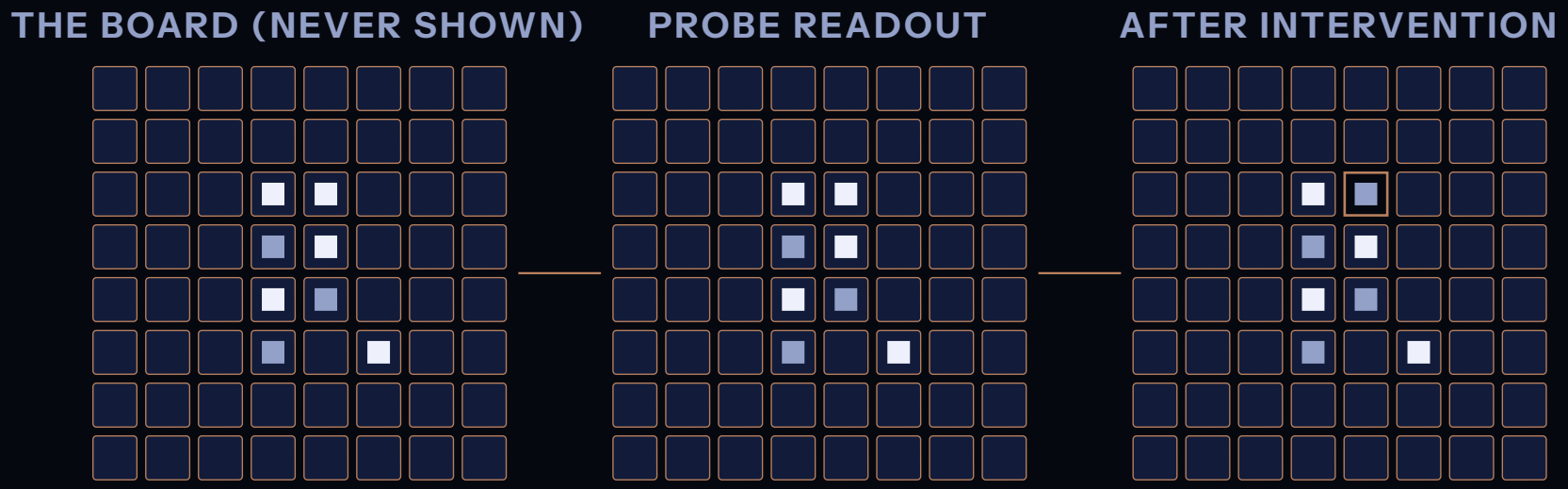
A forward pass has fixed depth. Each generated token is another pass, and nothing limits the count. A reasoning trace is the model running a procedure to completion.

Scale keeps working

Loss falls as a power law over seven orders of magnitude of compute. Architecture details matter far less than scale.

Kaplan 2020; Hoffmann 2022 · attention and pre-training: Oct 5.

Representations are the by-product



A model trained only on move sequences keeps a linear map of the board, and editing it changes the moves.

The cheapest way to predict text about the world is to represent the world.

Li 2023; Nanda 2023; Gurnee & Tegmark 2023 · interpretability: Dec 7.

The same bet at every stage

PRE-TRAINING	FINE-TUNING	RL ON CHECKED ANSWERS	RL ON TOOLS
representations	format, instructions	self-checking, long reasoning	strategies that transfer

Each stage sets a hard, varied objective with no way to memorize. Reasoning appeared under RL without being shown; tool skills learned on one domain transfer to others.

Brown 2020; DeepSeek-R1 2025; RITE 2025 · finetuning and RL: Oct 19.

Predicting text is not acting.
Post-training taught the model
to follow instructions, reason, and call tools.
Now it needs somewhere to act.

That is the harness.

Where we are

How they work

Context in, a distribution out, one draw appended. Tool results return text into the context.

Part 1 · 12 min

Why they work

Varied text, no memorizing, a hard objective: general strategies.

Part 2 · 8 min

The harness

Everything around the model. You build it and you control it.

Part 3 · 14 min

Working with them

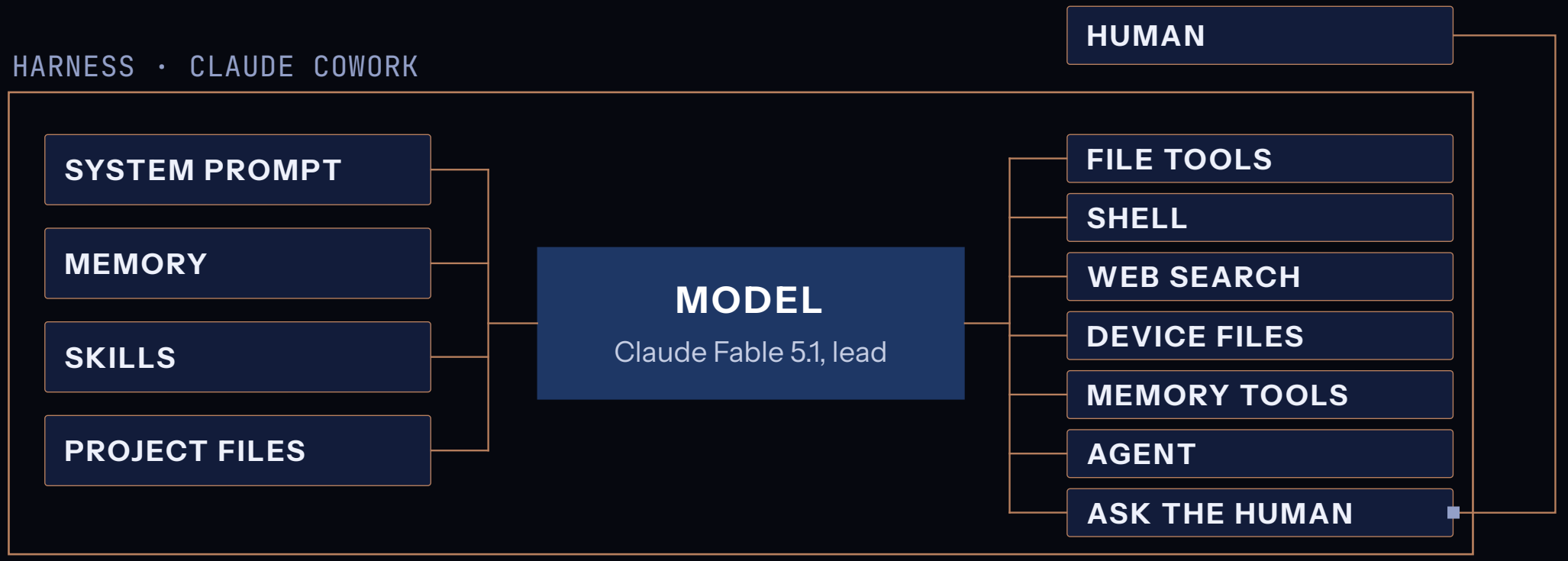
Clear vision, clear communication, clear execution.

Part 4 · 8 min

The harness assembles the prompt

Before you type, the harness has filled most of the context. It decides what goes in, in what order, which tools exist, and who approves an action.

The scaffold that built this talk



Everything in this diagram is real. The model is one box. The rest is the harness.

The files it drew on

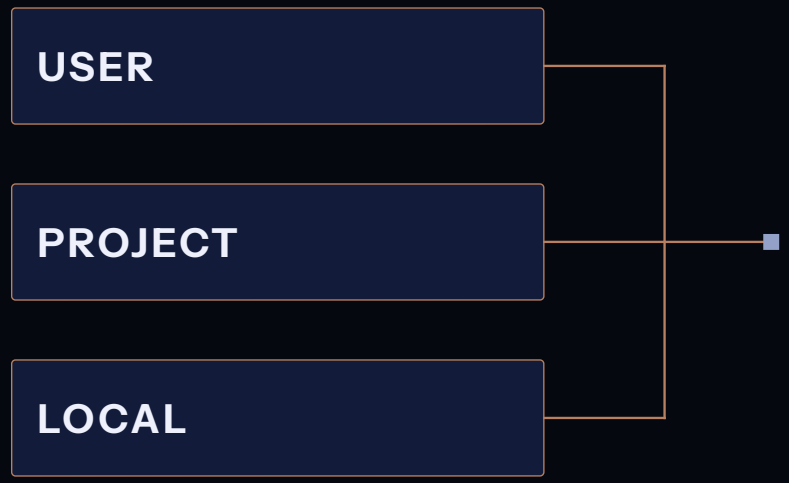
WHAT THE AGENTS READ AND WROTE

```
TALK_DESIGN.md
SESSION_ARTIFACTS.md
research/web_brief.md
research/why_they_work_brief.md
research/own_decks_brief.md
research/harness_brief.md
design/SLIDE_DESIGN.md
slides/lib/speagle_slides.py
slides/lib/README.md
build/build_stig.py
memory: profile, preferences, talks.md
```

Specs, research, the slide library, the builder script, and memory. The agents read these; you are looking at what they produced.

Instruction files

APPENDED EVERY SESSION

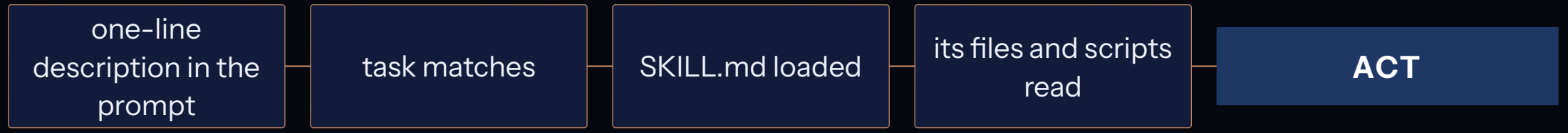


SLIDE_DESIGN.MD · §8 NEVER

“Never: emoji, icons standing in for words, drop shadows … more than one pad … two ideas on one slide.”

Standing text the harness appends at the start of every session: your rules, standards and specs. The spec behind these slides is one.

Skills load when needed

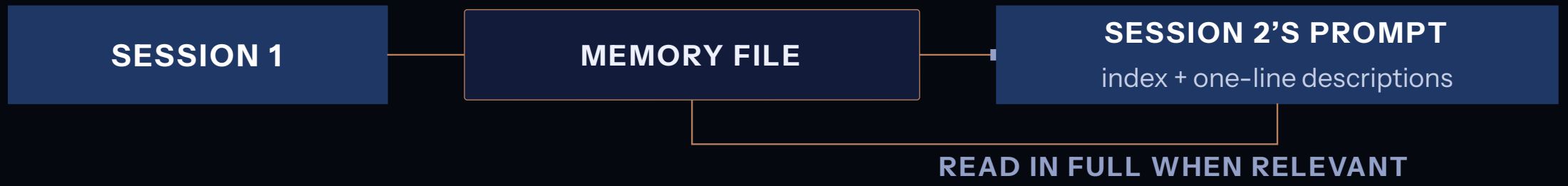


THE DESCRIPTION IN THE PROMPT · SKILL: PPTX

“Use this skill any time a .pptx or .potx file is involved in any way — as input, output, or both: creating slide decks … reading, parsing or extracting text … editing or updating existing presentations …”

Only the description sits in the prompt. When a task matches, the file is loaded and tells the model what else to read.

Memory is read the same way



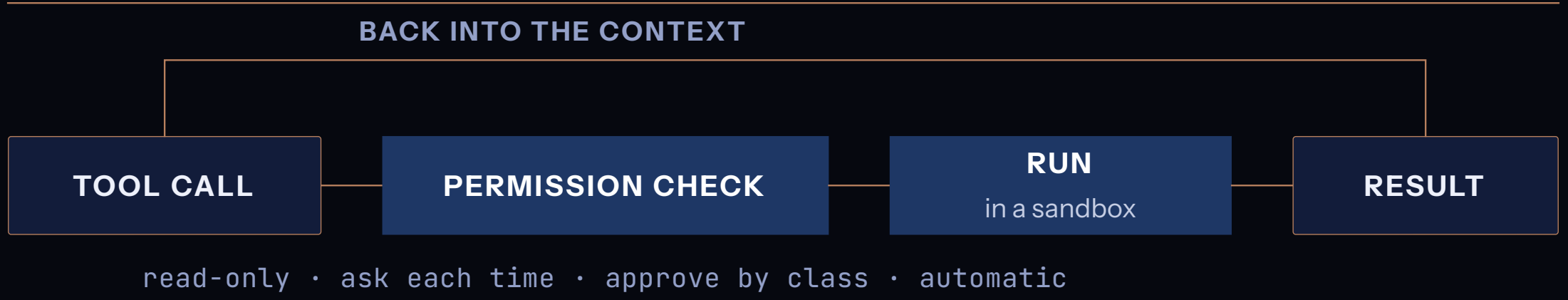
[stated] Slide design preference: integrated visual + textual representation on many slides; figures must reinforce and explain the point

[stated] Opening talk of the NASA Cosmic Origins AI/ML STIG fall 2026 lecture series (Mon Sep 14 2026, 4pm ET, Teams), ~50 min overview

The index is in every prompt; a file is read only when the task calls for it. This is how the same preferences reached every agent tonight.

`Retrieval and memory: Oct 26.`

Tools and permissions



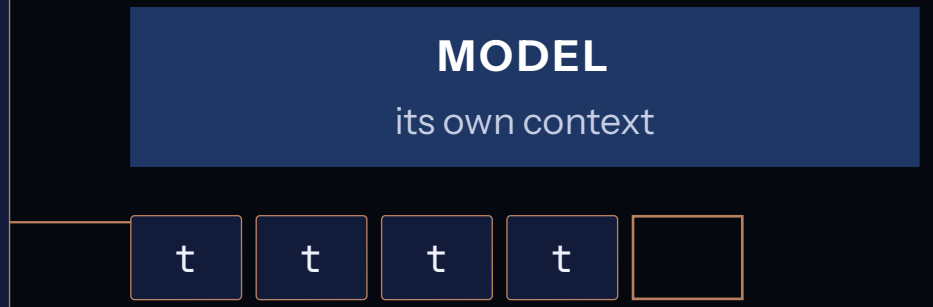
Every call passes a check the harness owns, from asking about everything to asking about nothing. MCP is the open standard for plugging tools in.

Tool use and MCP: Nov 2.

An agent can be a tool

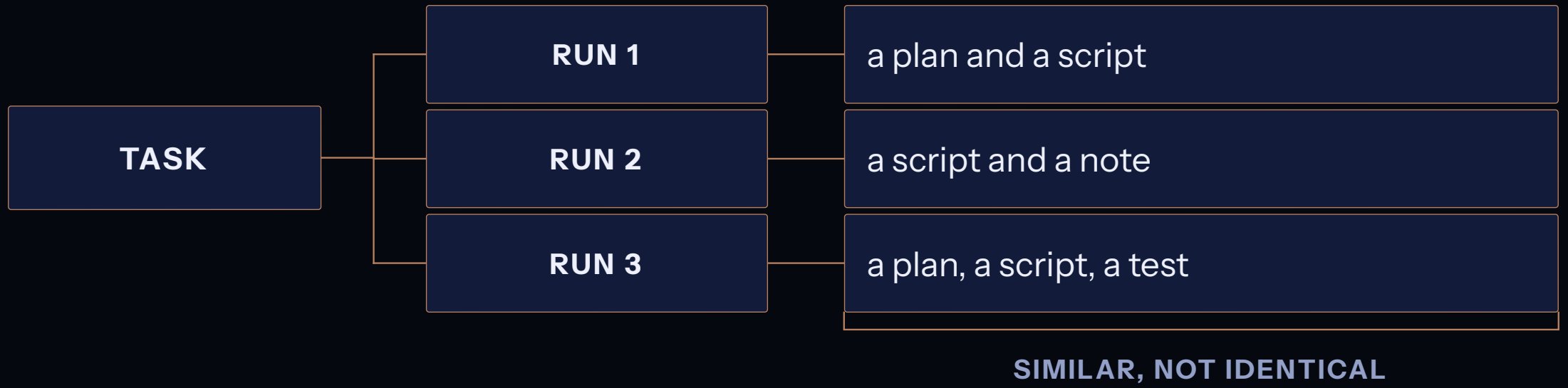
A TOOL DEFINITION

```
name: Agent
description:
  Launch a new agent to handle multi-step tasks
input:
  prompt   string
  model    string
returns: the agent's final message
```



Same shape as Read or Bash. The "code" that runs is another model with its own context; what comes back is text.

The same task, run three times



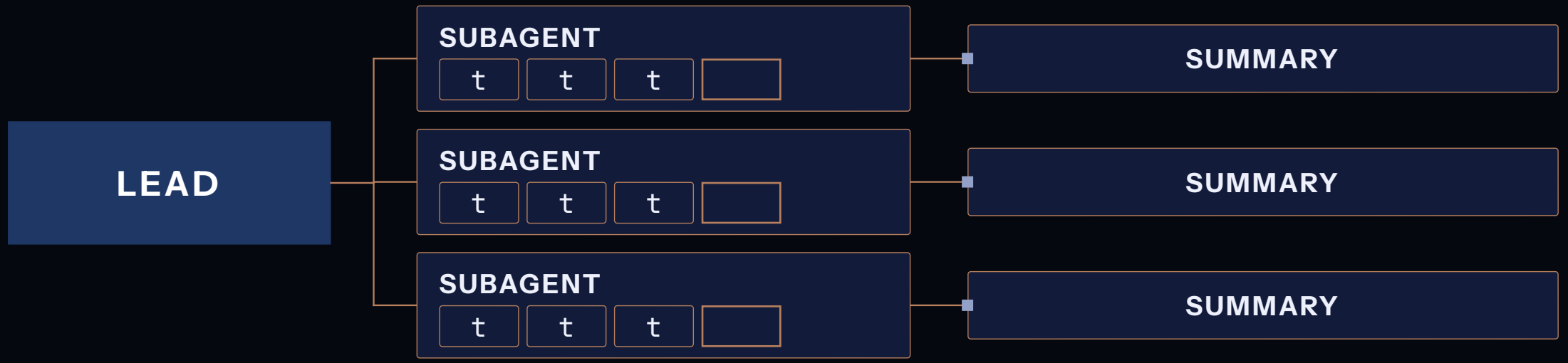
Outputs are drawn, not computed. Three runs land in similar places, never the same place, and you cannot reproduce one exactly.

Small changes in context, different behaviour

The model is the same. Change one line of the context and the behaviour changes.

That is a risk in one long conversation and a design tool in many short ones.

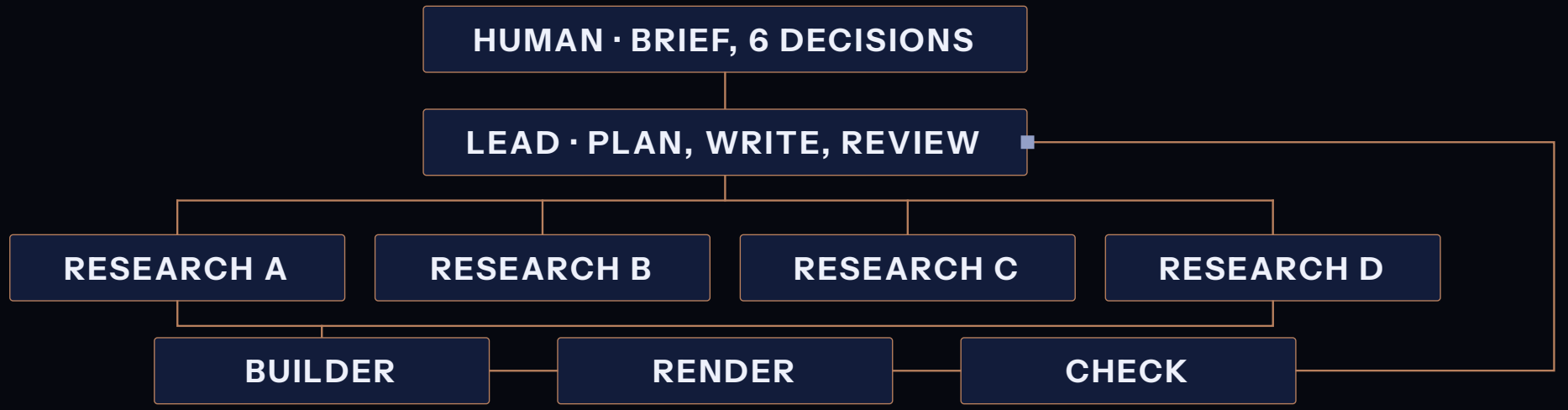
Why split the work across agents



Each agent gets a clean, purpose-built context and returns a summary. Work runs in parallel, detail stays out of the lead's window, and each result can be checked.

Cost: ~4× the tokens of a chat; multi-agent ~15× (Anthropic 2025) · orchestration: Nov 16.

The workflow that built this talk




```
research A · 2,472 words · 81 tools · 7.2 min     builder v1 · 44 slides · 67 tools · 17.4 min
research B · 1,900 words · 35 tools · 6.8 min     builder v2 · 46 slides · 107 tools · 24.5 min
research C · 1,484 words · 16 tools · 2.1 min     builder v3 · 46 slides · 94 tools · 21.4 min
research D · 1,700 words · 85 tools · 11.0 min    builder v4 · 55 slides · 120 tools · 25.7 min
```

Cheaper models did the implementation; the lead planned and wrote; the human decided. That rule was written down before anything ran.

When a result says no

THE SHELL, EVERY ATTEMPT

“sandbox-helper: no Plan9 drive shares mounted … A Windows update released September 8 prevents Claude’s workspace from reaching your files.”

Recovery: copy the files up, edit, copy back; the 278-file reorganization became a script Josh ran himself.

THE SCRIPT’S FIRST RUN

Moved 29 of 278 files, then stopped. The log ended at “OUPHZ_frrf_05-18-2016.pptx”.

Diagnosis: Dropbox held the log open. Fix: log in memory, continue past single failures. Second run: 249 moved, 0 failed.

Two tool results tonight were failures. Both were read and worked around. A check step is what turns a loop into a workflow.

Same model, different harness, different agent.
The harness is the part you control.

Prompt, tools, permissions, memory, skills, agents: each is a decision.

Where we are

How they work

Context in, a distribution out, one draw appended. Tool results return text into the context.

Part 1 · 12 min

Why they work

Varied text, no memorizing, a hard objective: general strategies.

Part 2 · 8 min

The harness

Everything around the model. You build it and you control it.

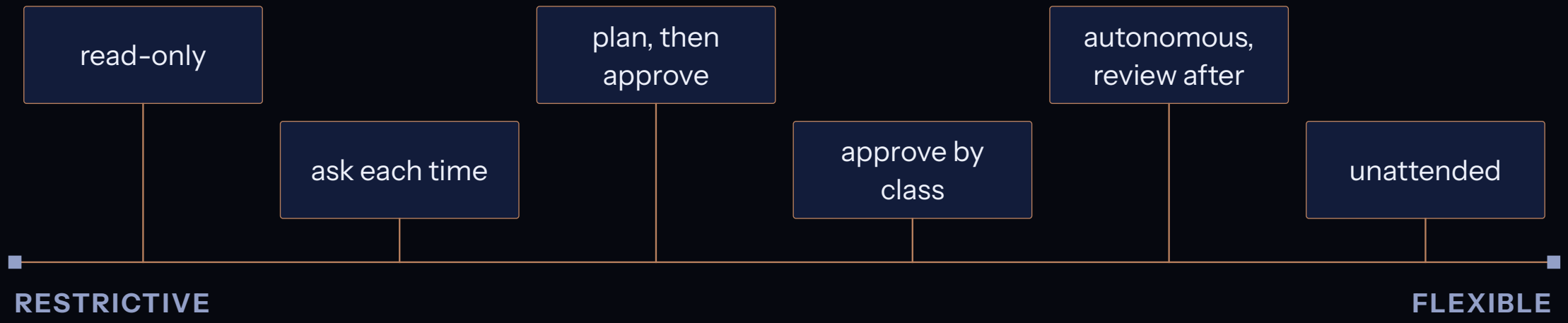
Part 3 · 14 min

Working with them

Clear vision, clear communication, clear execution.

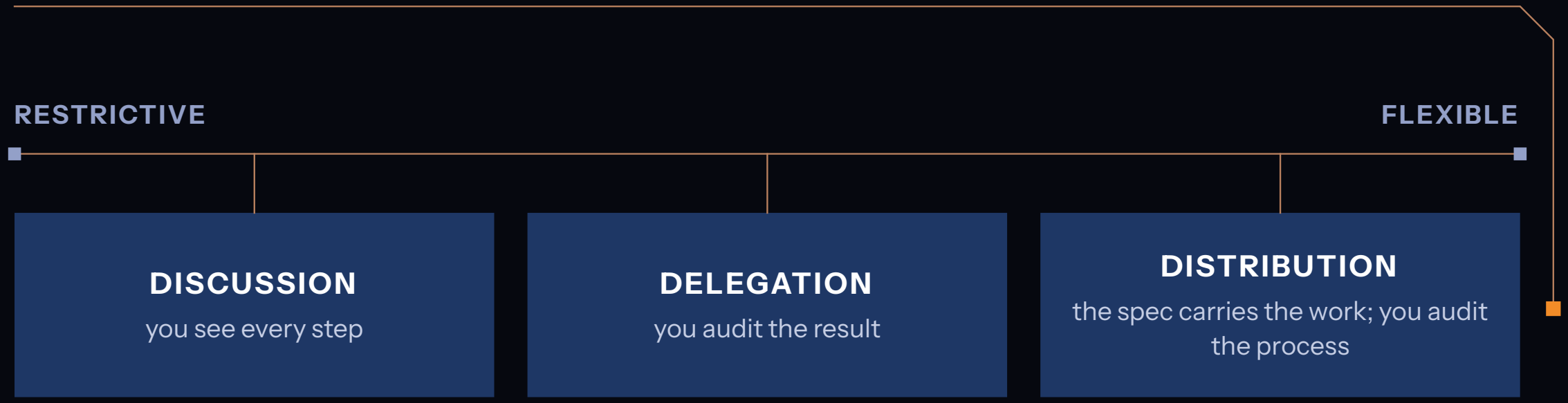
Part 4 · 8 min

Restrictive to flexible



One workflow moves along this axis: restrictive while setting up, flexible once specified, restrictive to check. It shifts as models and your work change.

Discussion, delegation, distribution



The further right, the less you see and the more the specification has to carry.

Vision, communication, execution

CLEAR VISION
know what you want

CLEAR COMMUNICATION
say it so someone could act on it

CLEAR EXECUTION
then it executes

YOURS

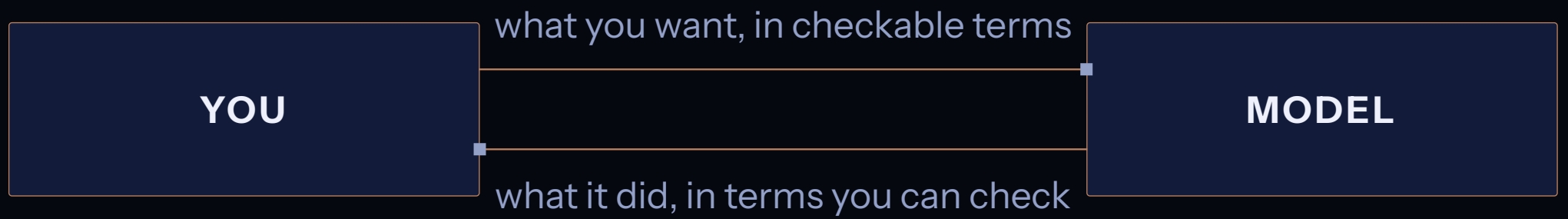
THE AGENT'S

Know what you want. Say it so an agent, or a colleague, could act on it without you. The first two are most of the work; do them first.

Intent into things the model reads

Anything every run should respect goes in a file the harness loads. Written once, read every time. Keep it short and prune what stops being true.

Communication runs both ways



Ask for reports you can verify: assumptions stated, sources resolved, changes listed. Do not accept output you cannot read.

What it costs to skip this

Developers with early-2025 AI tools were 19% slower and believed they were 20% faster (METR 2025).

People who shipped code with AI scored 50% on understanding it, against 67% without (Anthropic 2026).

Treated like ordinary code, these systems cost time. Treated as systems you design and monitor, they return it.

METR, Jul 2025 and Feb 2026 · Anthropic, Jan 2026

Be the showrunner, not the audience.

After John Wu, “Clear Vision, Clear Communications” (2025), quoting Javier Grillo-Marxuach, The Eleven Laws of Showrunning.

How long a task can an agent finish

METR measures the human working time of coding tasks an agent completes half the time. Doubling about every seven months since 2023.

Caveats: coding only; the top bar's interval is 2–20 hours · METR Jan 2026.

What agents did for this talk

TASK	WHO · HOW LONG
Reorganize 270 files, 3.7 GB	script, run by the human
Index 176 talks	script
Research, four briefs	4 agents, 2–11 min each
Build the slides, four versions	4 agents, 17–26 min each
Plan and every line of text	lead
Scope, structure, what to cut	human, 6 checkpoints

The unit of work moved from lines of code to whole tasks. The human’s work moved to the start and the end.

Four things to take away

How they work

Context in, distribution out, one draw appended. Tool results are text the model did not write.

Why they work

No memorizing, varied text, a hard objective. Representations are the by-product.

The harness

Prompt, tools, permissions, memory, skills, agents. You choose where it sits on the axis.

Working with them

Vision, communication, execution. Intent into files. Monitor the harness.