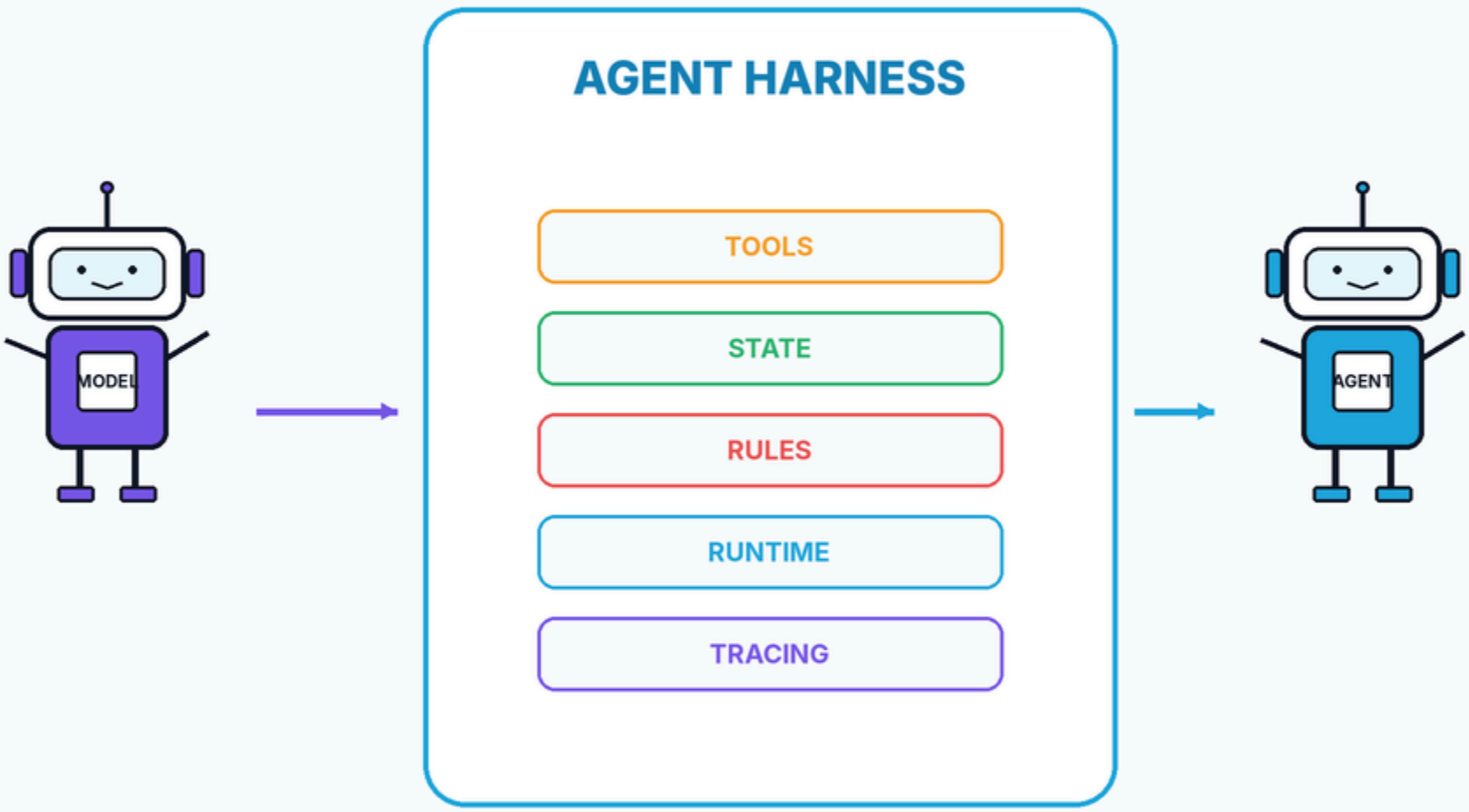
WHY CAN THE SAME MODEL FEEL LIKE A COMPLETELY DIFFERENT AGENT?

Meet the Agent Harness — the runtime around the model that turns intelligence into dependable work.



The model supplies intelligence. The harness supplies the operating system around that intelligence.

THE MODEL ALONE IS NOT THE AGENT

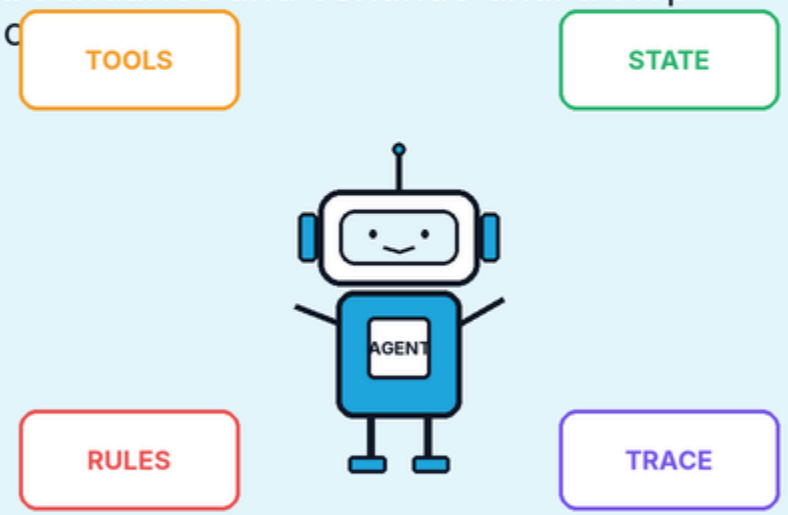
A model can reason and generate actions. Something else must manage the loop around it.

MODEL ONLY

Can propose a tool call, but does not automatically execute it, persist state, enforce permissions or recover from failure.

MODEL + HARNESS

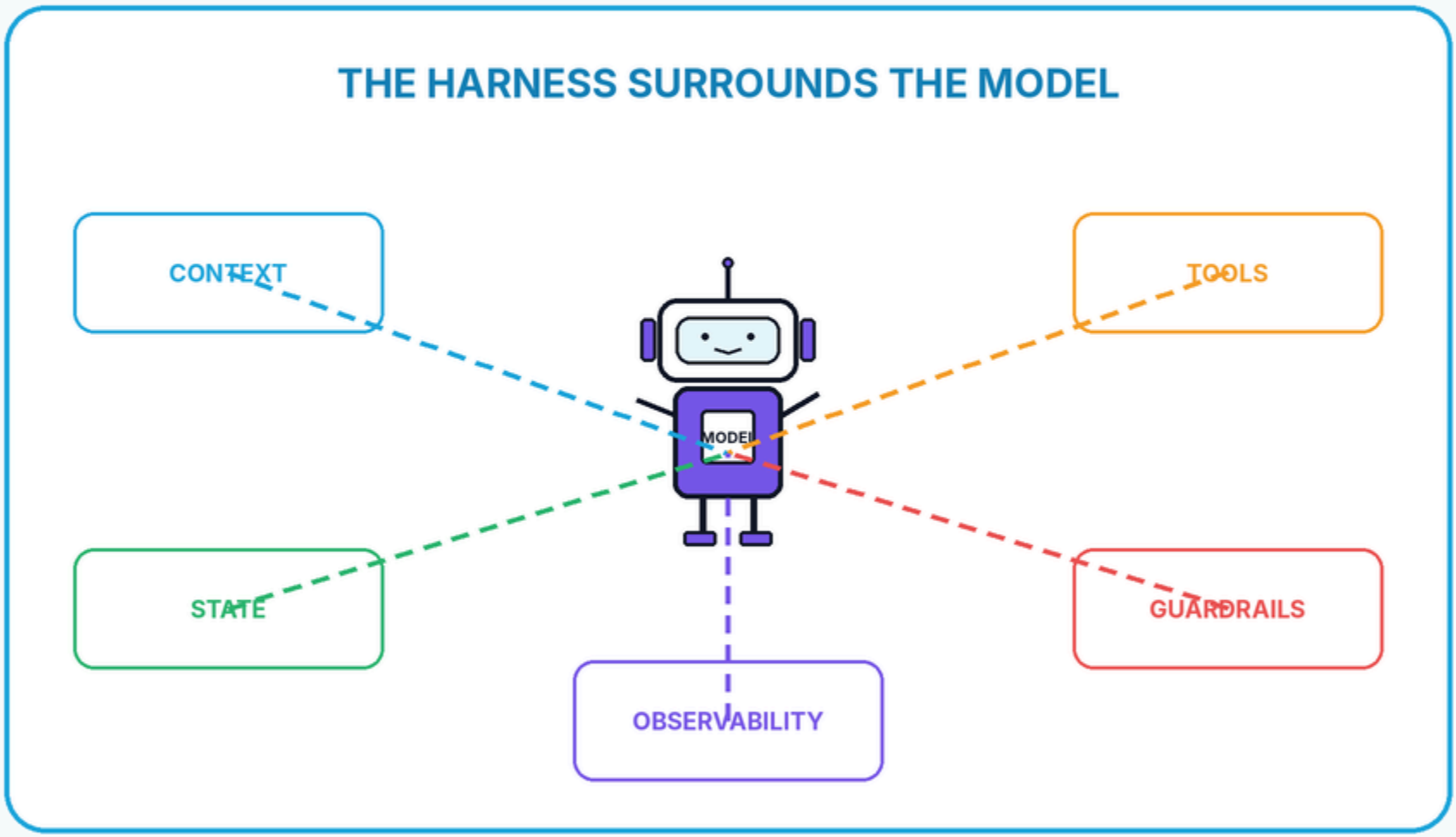
Can reason, act through tools, observe results, remember progress, obey boundaries and continue until a stop c



A capable model is a component. An agentic system is model + runtime + environment + controls.

WHAT IS AN AGENT HARNESS?

It is not one formal protocol. It is the practical scaffolding that runs and controls an agent.



Think of “harness” as the execution and control layer that makes the model usable as an agent.

THE EASIEST ANALOGY: ENGINE VS VEHICLE

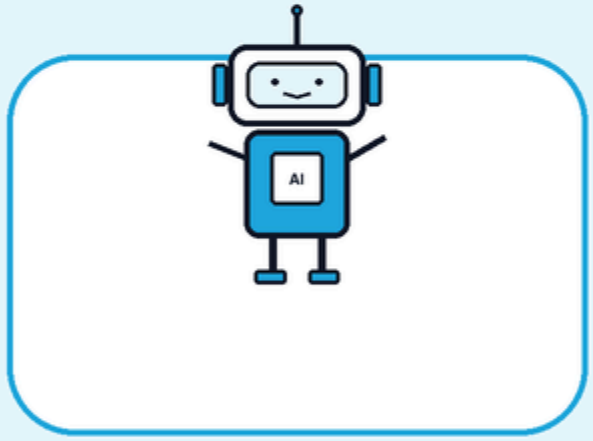
A powerful engine is useful only when the rest of the system lets it move safely and reliably.

MODEL = ENGINE

Provides raw capability: reasoning, language, planning and tool selection.

HARNESS = VEHICLE

Adds controls, instruments, brakes, fuel, navigation and the rules for operating the engine.



STATE

TOOLS

RULES

Better agent performance often comes from the whole system — not only from swapping the model.

HARNESS VS FRAMEWORK: DO NOT CONFUSE THEM

A framework gives you building blocks. Your harness is the configured system you actually run.

AGENT FRAMEWORK

A library or SDK: runners, tool abstractions, sessions, tracing APIs, handoffs and more.

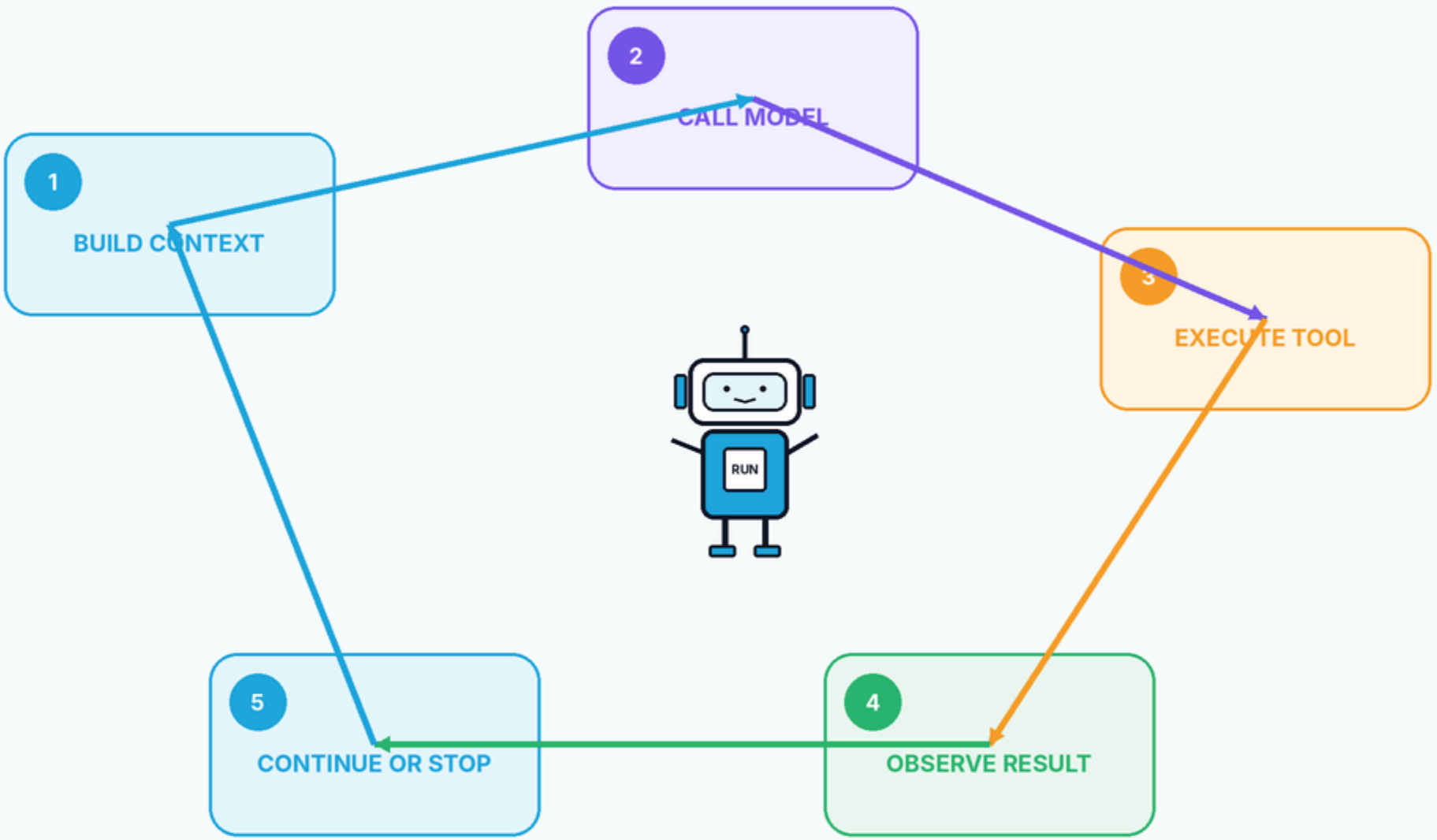
YOUR HARNESS

The concrete combination of model, prompts, tools, permissions, state, environment, retries, approvals and observability.

OpenAI Agents SDK, Claude Agent SDK, LangGraph and others can help you build a harness — they are not your entire harness.

THE CORE AGENT LOOP LIVES IN THE HARNESS

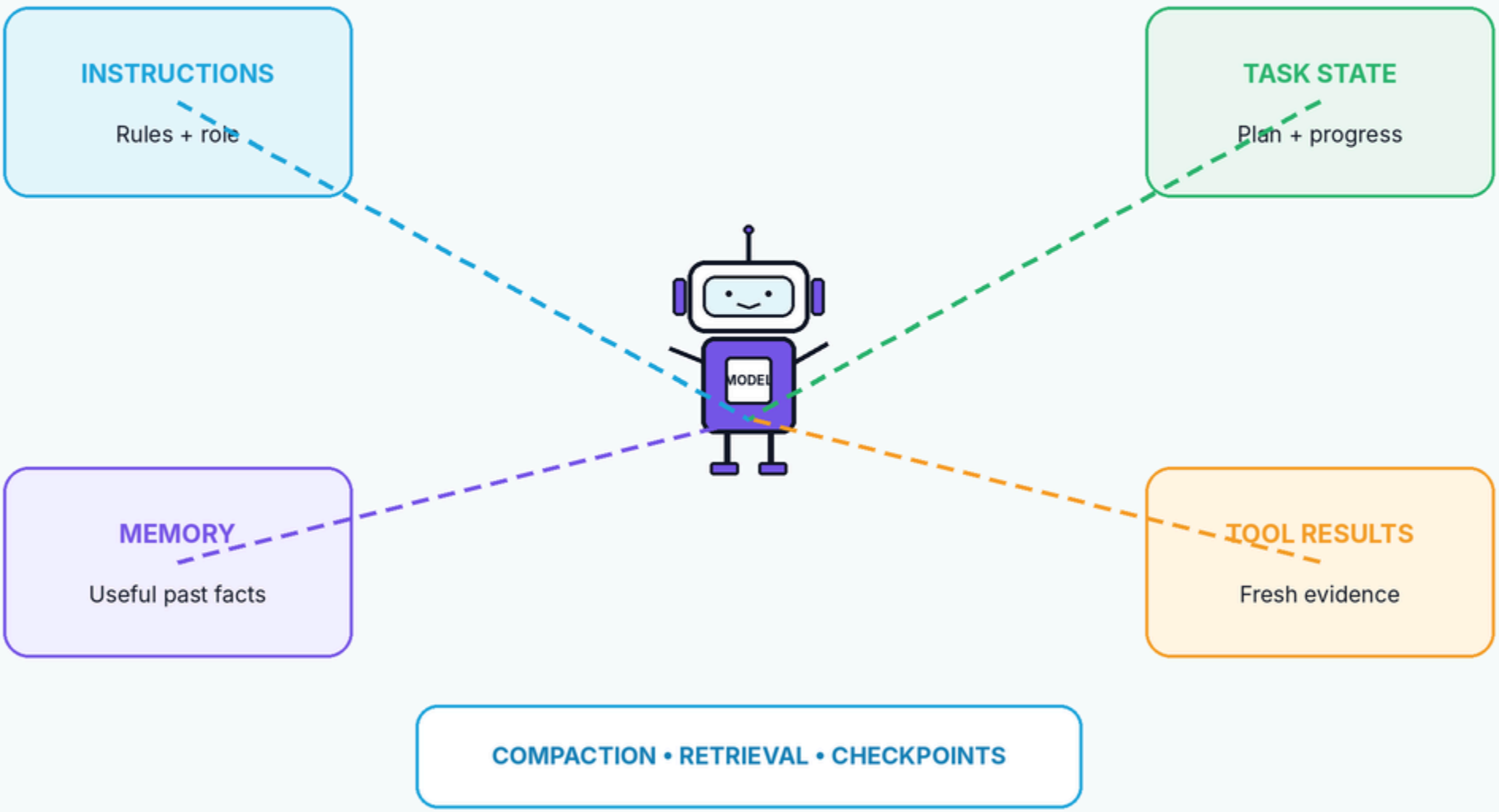
The harness decides how model turns, tool execution and observations are stitched together.



The harness owns the loop: prepare → reason → act → observe → repeat — until a stop condition is reached.

CONTEXT + STATE: WHAT SHOULD THE MODEL SEE NOW?

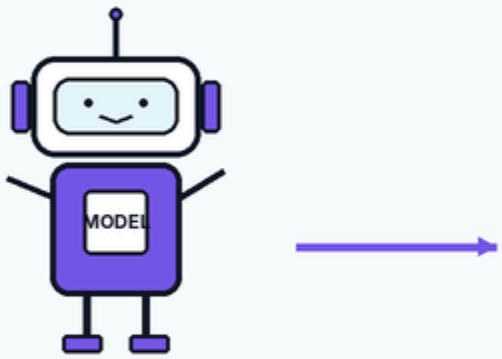
The harness continually decides what information enters the next model call — and what stays outside.



Context engineering is a harness job: the next decision is only as good as the working view you construct.

TOOLS NEED AN EXECUTION LAYER

The model can request an action. The harness decides whether, where and how that action actually runs.



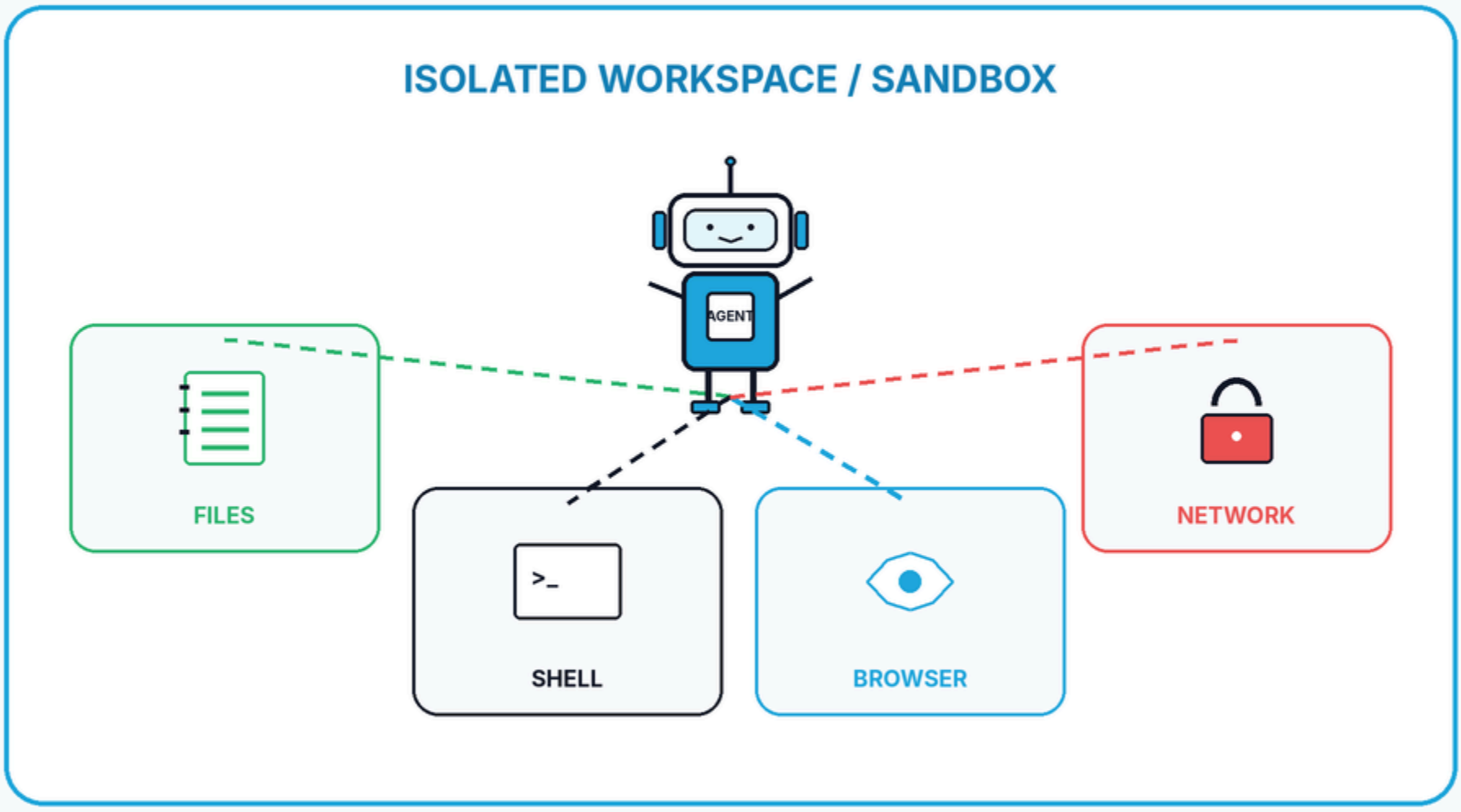
TOOL EXECUTION

1. VALIDATE ARGUMENTS
2. CHECK PERMISSION
3. RUN TOOL
4. HANDLE TIMEOUT / ERROR
5. RETURN STRUCTURED RESULT

A tool call is not “magic model action.” It is a request passing through runtime checks and execution code.

WHERE DOES THE AGENT ACT? MEET THE WORKSPACE

Some agents need a real environment: files, shell commands, browsers, containers or isolated sandboxes.



The harness defines the agent's "hands" — and the boundaries of the environment those hands can touch.

GUARDRAILS + PERMISSIONS: WHERE MUST THE AGENT STOP?

A production harness places policy around inputs, outputs and especially consequential tool calls.

ALLOW AUTOMATICALLY

Low-risk reads, calculations, searches or reversible actions can run inside predefined limits.

BLOCK OR ASK

Payments, destructive writes, sensitive data access or policy violations may require rejection or human approval.

Autonomy is not “let the model do anything.” It is permissioned action inside explicit boundaries.

RELIABILITY CONTROLS: WHAT HAPPENS WHEN THINGS FAIL?

Real tools time out, networks break, arguments are malformed and duplicate actions can be dangerous.

RETRY

Transient failure

TIMEOUT

Do not wait forever

ERROR PATH

Return useful failure

IDEMPOTENCY

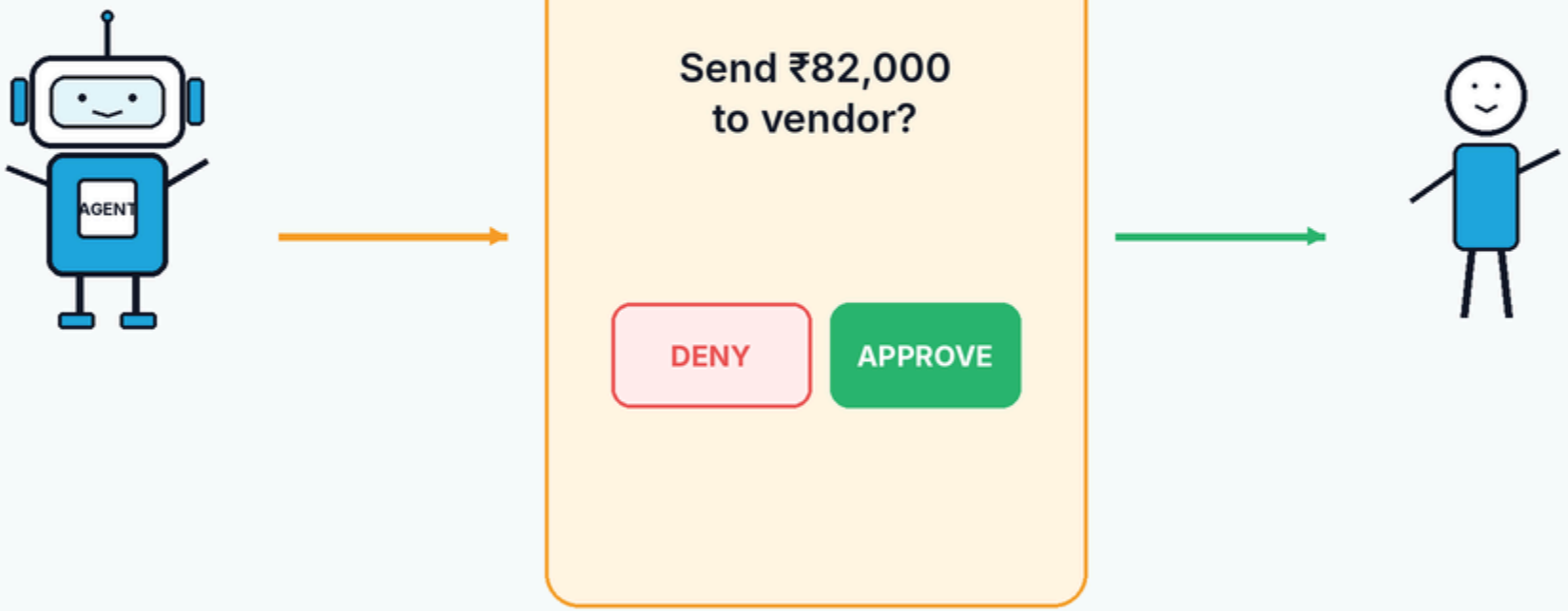
Avoid duplicate effects

FAILURE HANDLING BELONGS IN THE SYSTEM — NOT IN WISHFUL PROMPTING.

Reliable agents need deterministic runtime behavior around unreliable external systems.

HUMAN-IN-THE-LOOP IS PART OF THE HARNESS

The runtime can pause before a consequential action, preserve state, collect a decision and then resume.

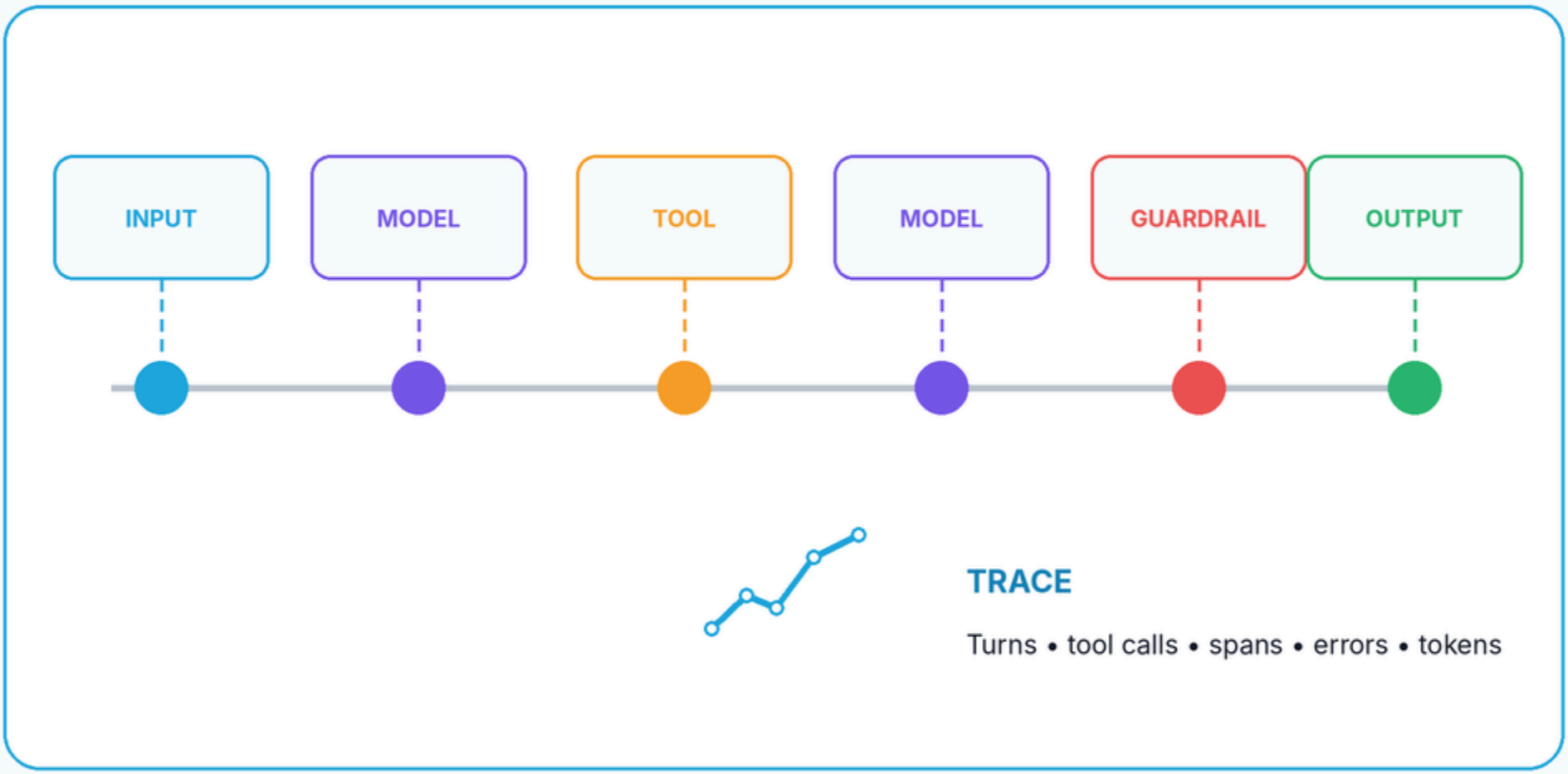


PAUSE → SAVE STATE → DECIDE → RESUME

Human approval is a runtime state transition — not just a sentence in the prompt.

TRACING + OBSERVABILITY: CAN YOU SEE WHY IT FAILED?

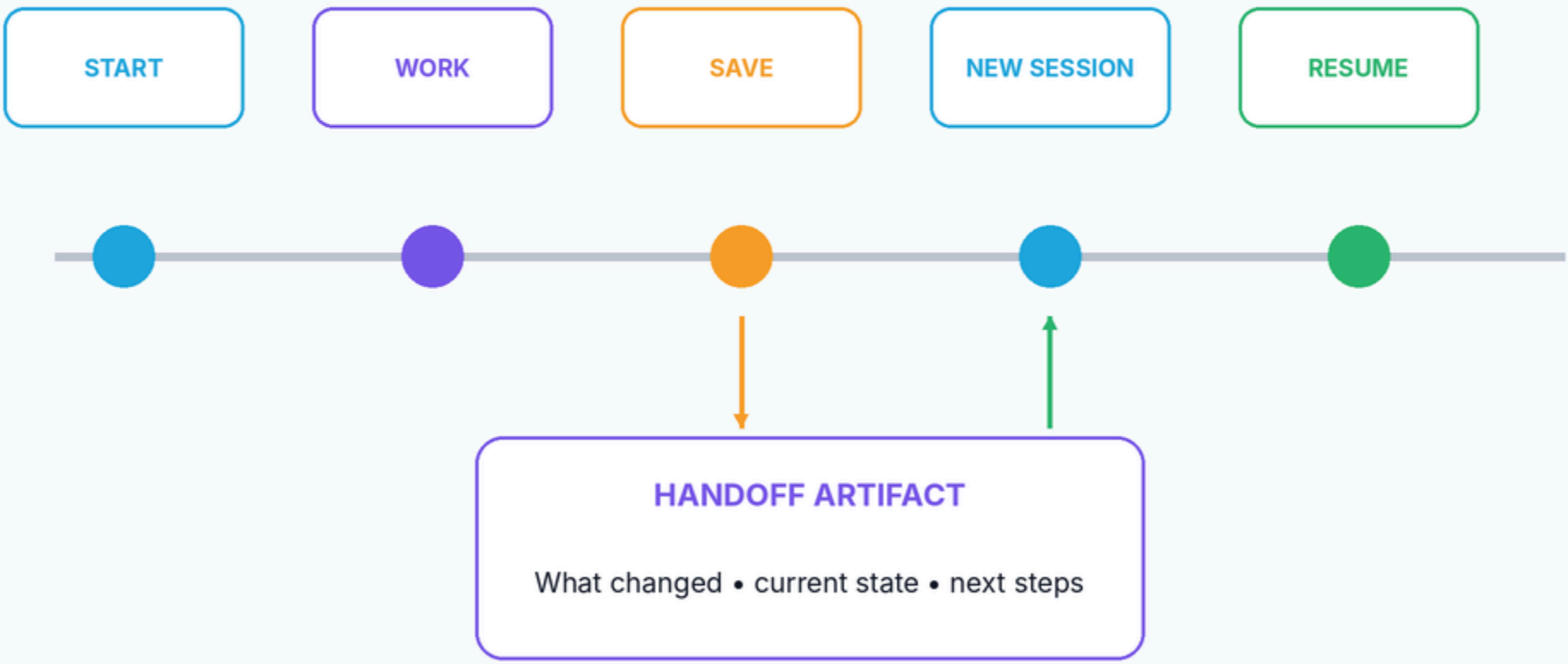
A harness should record enough of the run to debug behavior, cost, latency and tool usage.



Without traces, an “agent failed” report is vague. With traces, you can inspect the exact run path.

LONG-RUNNING AGENTS NEED A STRONGER HARNESS

When work spans context windows or sessions, the harness must preserve progress across boundaries.



Compaction, checkpoints and structured handoff artifacts let a fresh context continue old work instead of restarting.

ONE HARNESS CAN RUN ONE AGENT — OR A TEAM

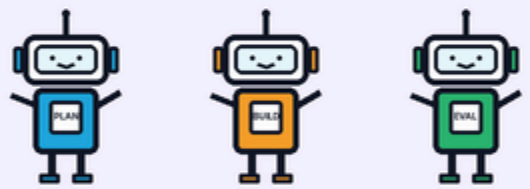
Multi-agent is an orchestration choice inside the harness, not a requirement for being agentic.

SINGLE-AGENT HARNESS

One decision-maker uses tools, state, guardrails and the runtime loop to finish a narrow job.

MULTI-AGENT HARNESS

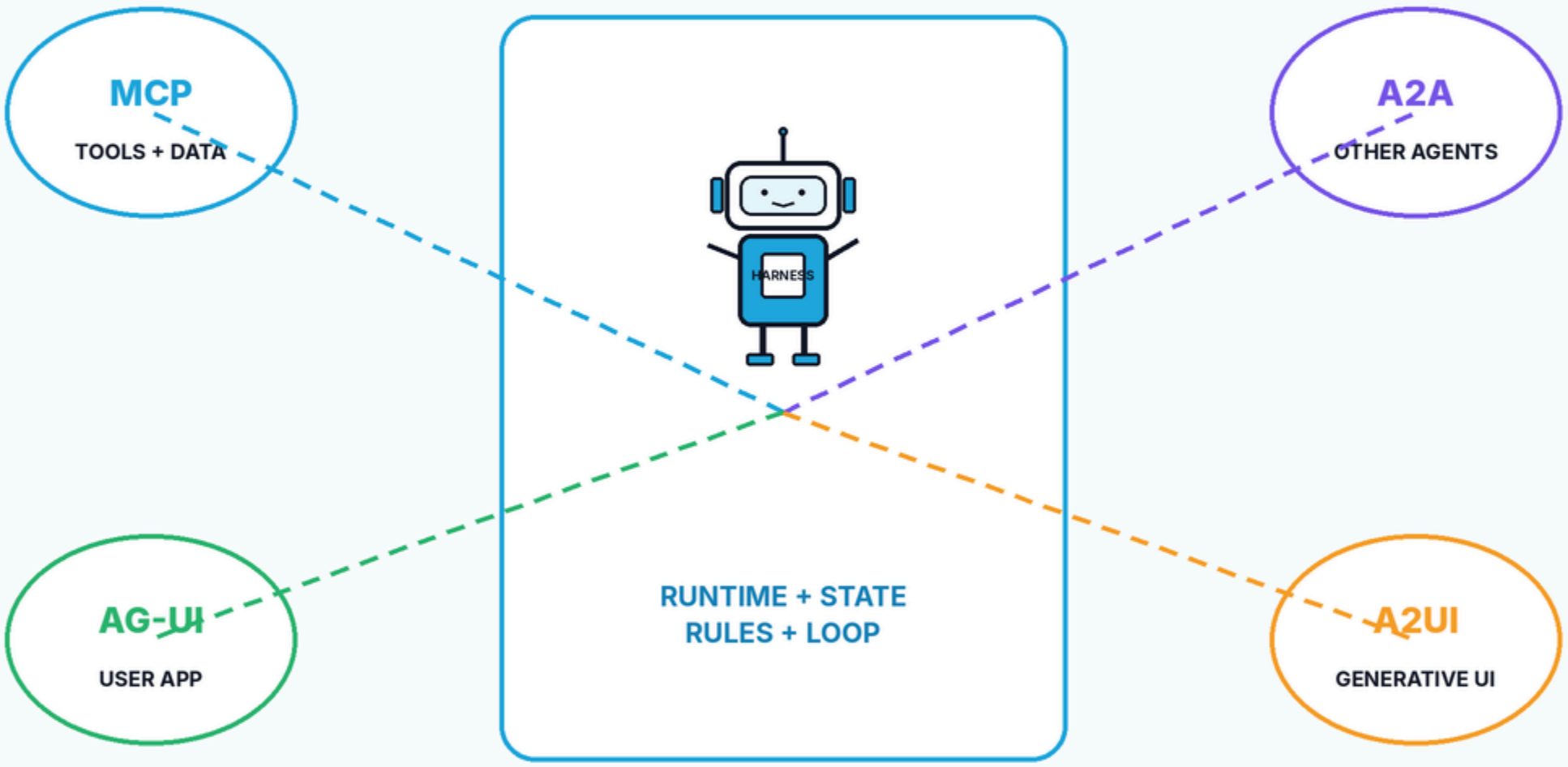
The harness can route work across planner, generator, evaluator or domain specialists — when that split adds value.



The harness coordinates agents, handoffs and shared state. More agents are useful only when specialization pays for the overhead.

WHERE DO MCP, A2A, AG-UI AND A2UI FIT?

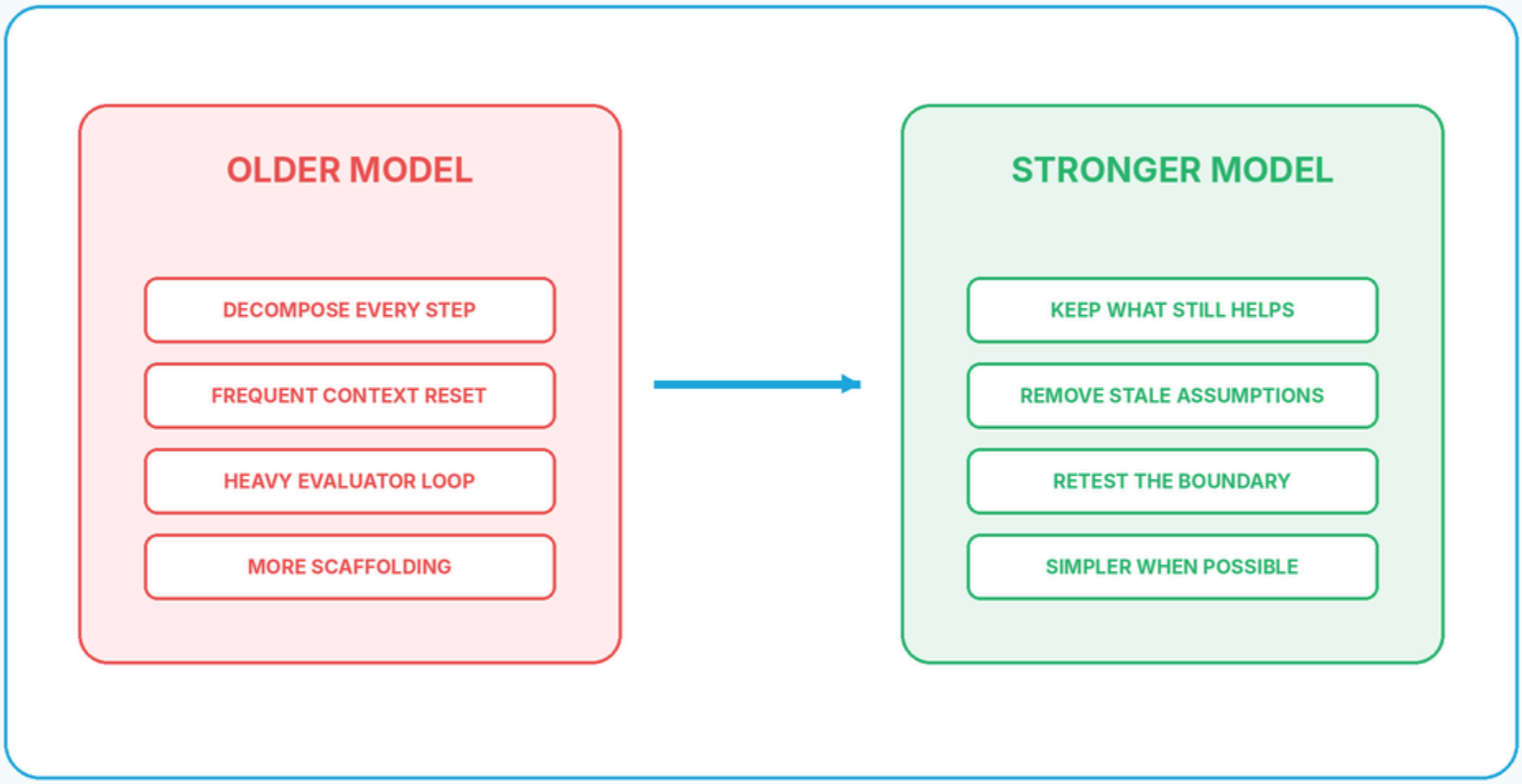
Protocols are connections the harness can use. The harness is the runtime that decides when and how to use them.



Protocols standardize interfaces. The harness owns orchestration, policy, state and execution around those interfaces.

A GOOD HARNESS EVOLVES WITH THE MODEL

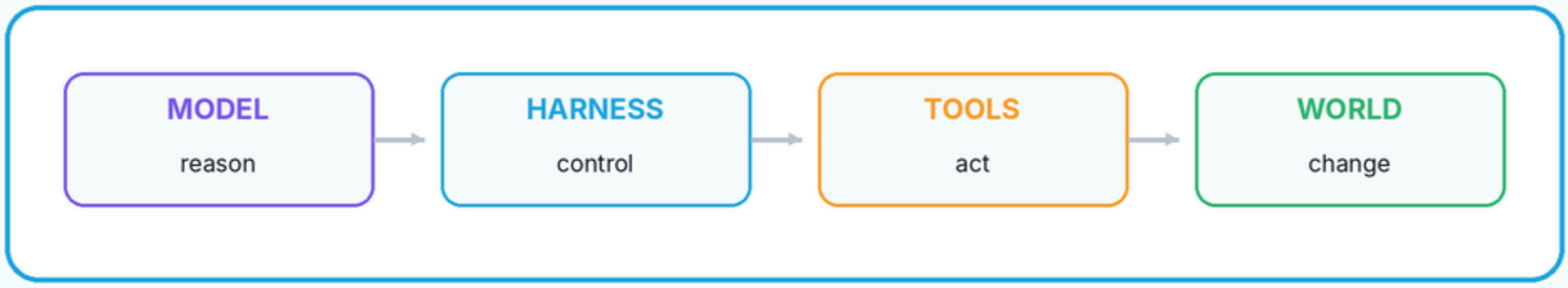
Scaffolding that helps one model may become unnecessary — or even harmful — as models improve.



Harness components encode assumptions about model weaknesses. Re-test those assumptions as capability changes.

DAY 20 TAKEAWAY

The agent harness is the control and execution system around the model.



LOOP

Turns reasoning into action

STATE

Keeps the right context

SAFETY

Controls permissions

TRACE

Makes behavior inspectable

A BETTER MODEL HELPS. A BETTER HARNESS MAKES THAT CAPABILITY OPERABLE.

NEXT: DAY 21 — Agent Evals: how do you know an agent actually works?

From “looks good” to repeatable tests, graders and task-level metrics.