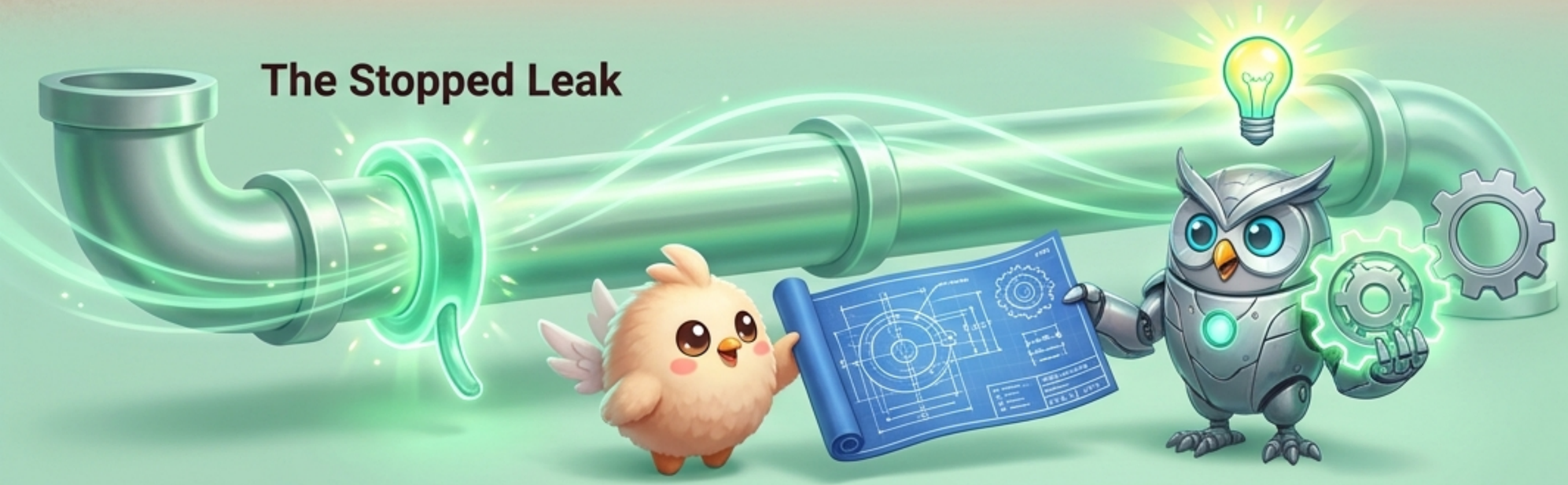
Before the Agent: The Pre-Flight Clarity Scorer

- Evaluating prompts before they reach the model
- Catching ambiguity early, asking targeted follow-ups, and learning user style over time

Most agent failures are *specification* failures, not capability failures

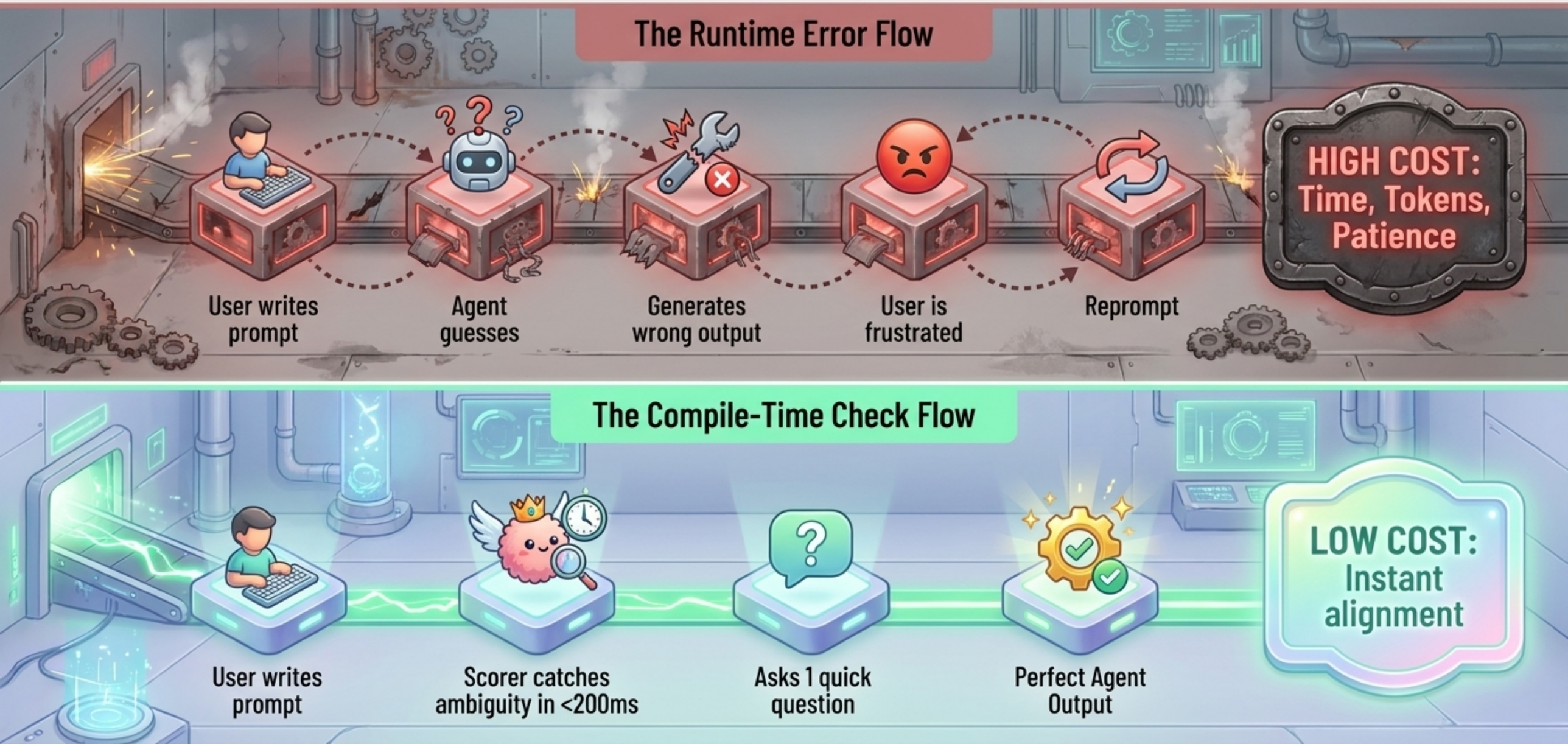


The Compounding Cost

- The agent does exactly what was asked, but not what was wanted.
- The user reviews, re-prompts ("No, I meant..."), and the agent re-does the work.
- Eventually, frustrated users just do it themselves.

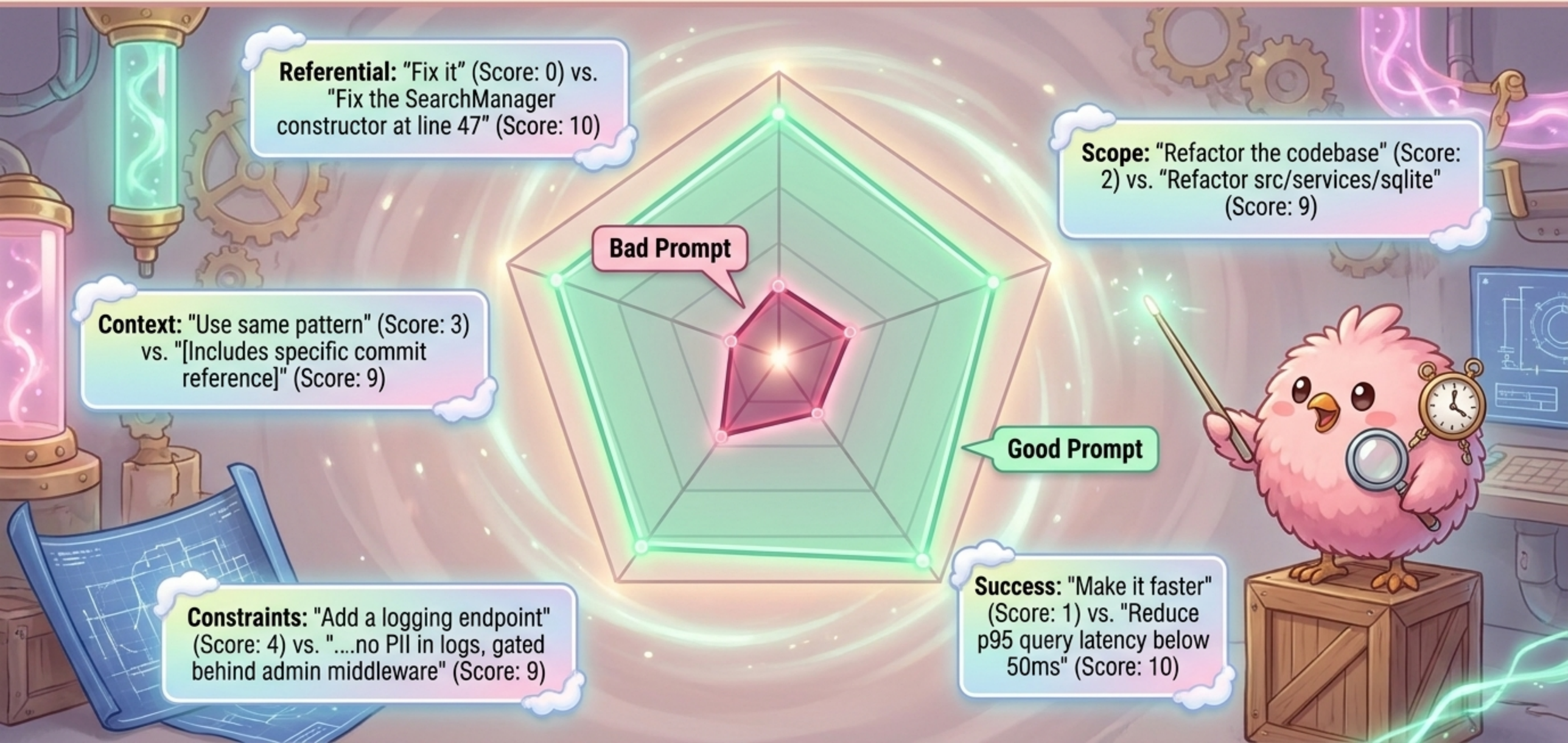
Shifting left: Why cheap upstream scoring beats expensive downstream regeneration

Catching bugs where they are cheap. A 200ms pre-flight check that catches even 50% of specification failures saves enormous amounts of downstream work.



Clarity is not a single number, but a vector across five dimensions

Composite scores are a weighted sum across specific failure modes.



The three-station assembly line of the Pre-Flight Workshop

The architecture relies on lightweight intervention, configurable thresholds, and continuous background learning.

1. Score: Micro-model evaluation

2. Gate: Configurable threshold routing

3. Learn: Background idiolect mapping

Station 1: The Scorer operates under a strict latency budget.

- Powered by a **small, fast model** (Haiku 4.5 or local fine-tune).
- This is the critical constraint: if the **pre-flight check is slow**, users will disable the tool entirely.

Station 2: The Gate rules determine when to interrupt

The gate is highly configurable. Power users tune for flow; new users opt for guardrails.

Score
> 7

Silent Pass-Through

Big Claude gets the prompt directly. No friction, zero interruption.

Score
5-7

Soft Preface Injected

The prompt proceeds, but Pompom attaches a silent note to the agent: “Note: I am interpreting X as Y — say if that is wrong.”

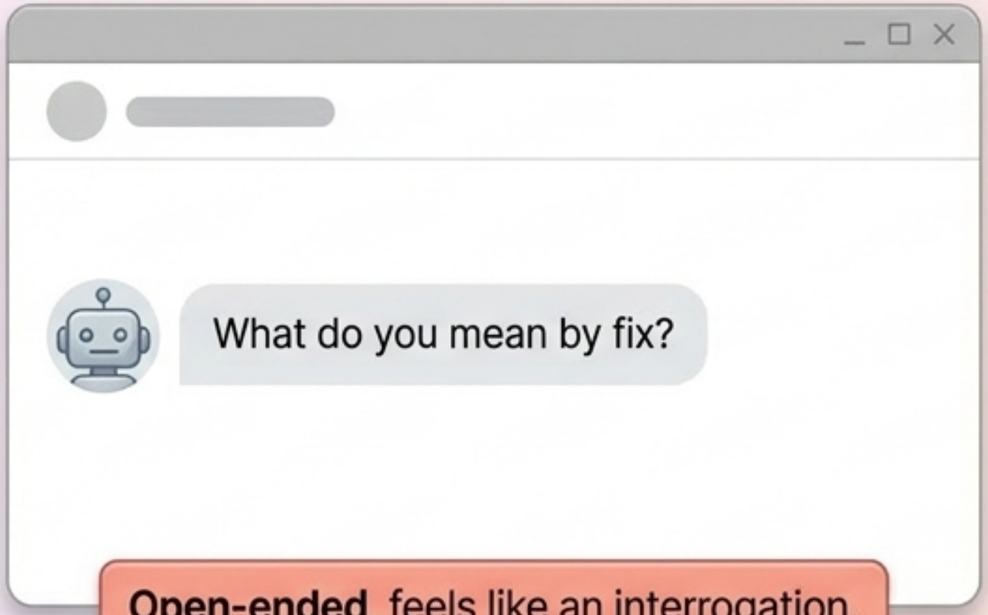
Score
< 5

Block & Ask

The prompt is paused. Surfaces a maximum of 2 specific questions via **AskUserQuestion** before running the main turn.

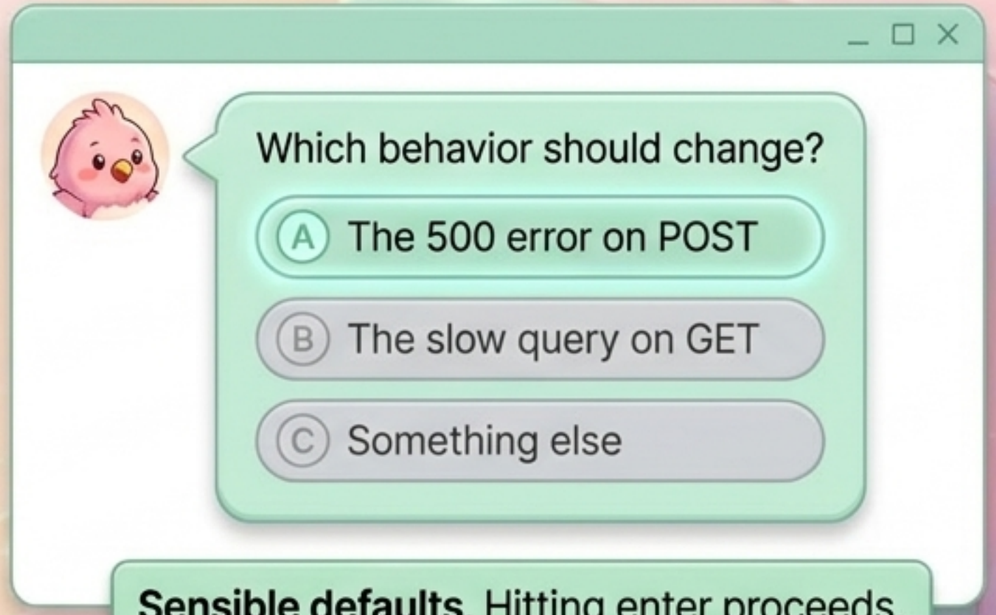
Designing the perfect follow-up: Clarifying without nagging

The Bad Way



Open-ended, feels like an interrogation.

The Good Way



Sensible defaults. Hitting enter proceeds.

- Max **two follow-ups** per prompt. Any more risks abandonment.
- Questions must offer **concrete options** with **sensible defaults**.
- The chosen answer is woven directly into the final prompt—Big Claude never sees the ambiguous version.

Station 3: The Learner turns every interaction into training data

We retroactively label whether the score was right based on what happens after the agent acts

User accepts first output

Positive signal

User re-prompts

Negative signal

“Actually I meant...” corrections

Labelling signal

Memory Jar

Clarity Radar Chart

Dynamically turns the weights of the clarity dimensions to custom user settings.

Over time, the scorer stops applying generic rules and learns the user’s personal idiolect.

From interrogation to symbiosis: Building a shared language

To a new system, “fix the thing” is hopelessly ambiguous. To a **personalized Scorer**, it’s a perfectly clear instruction.

Week 1: Interrogation

Week 3: Learning

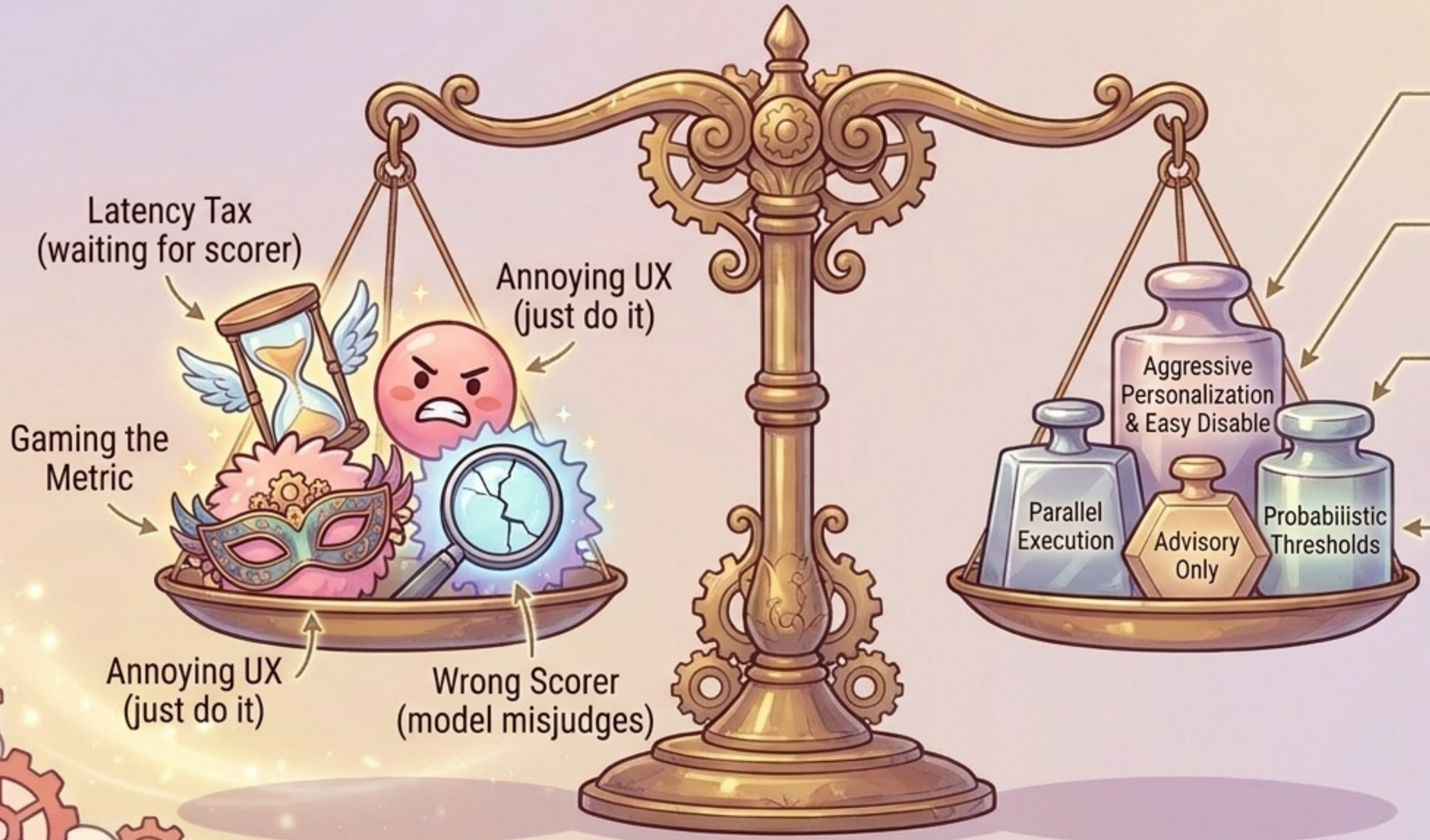
Week 8: Symbiosis

A natural fit for the existing claude-mem architecture

We already have the hooks, the cross-session context, and the observation pipeline. Personalization accumulates over months.

Anticipating the failure modes

The system is designed defensively. Asking a quick question is always cheaper than a full regeneration.



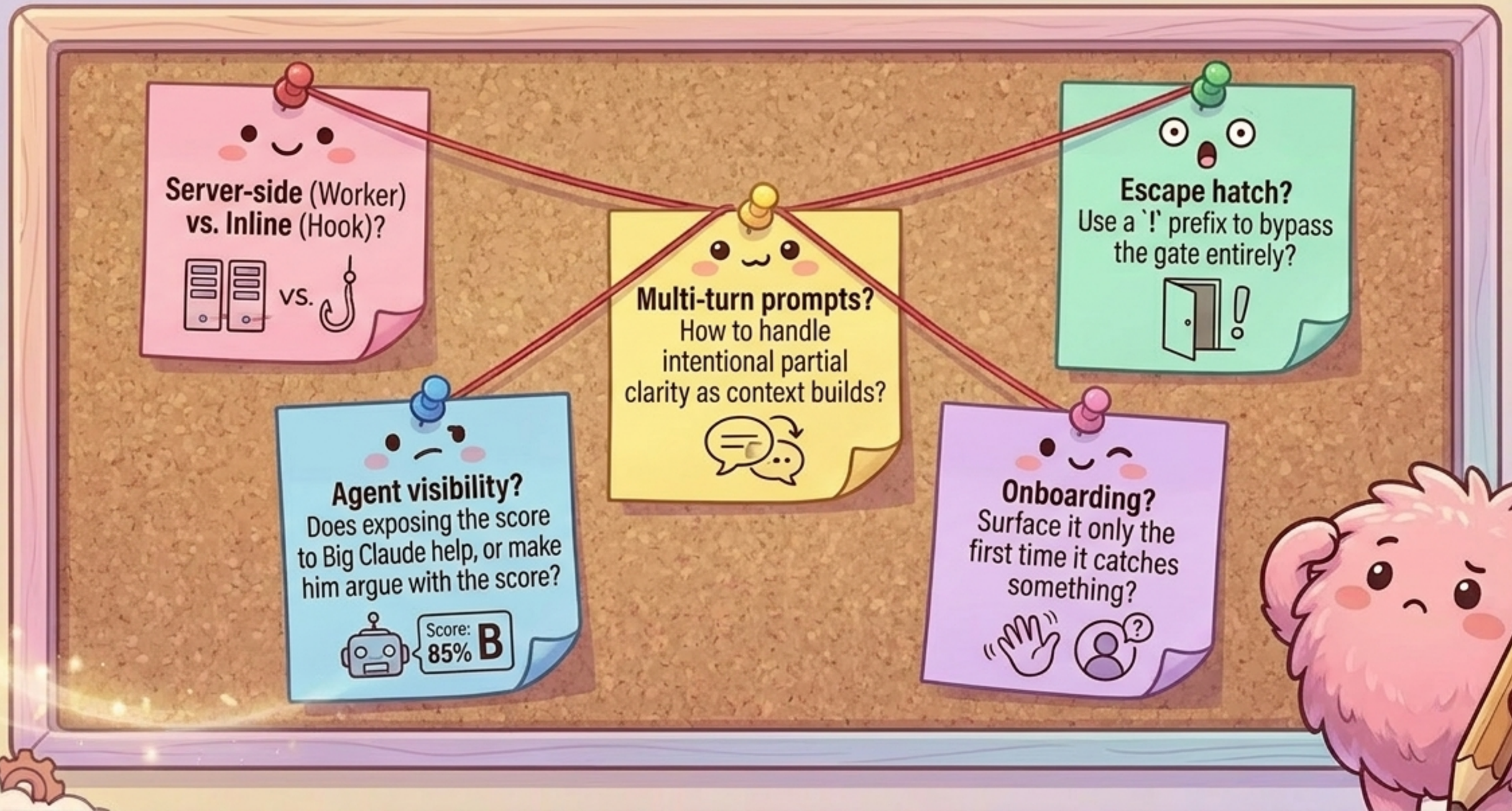
- Only block if low score returns before main turn.
- Tune for zero friction.
- Metric reflects downstream success, never forced.
- Tune for false positives (costs one question) over false negatives.

The Minimum Viable Test: A two-week plan for signal

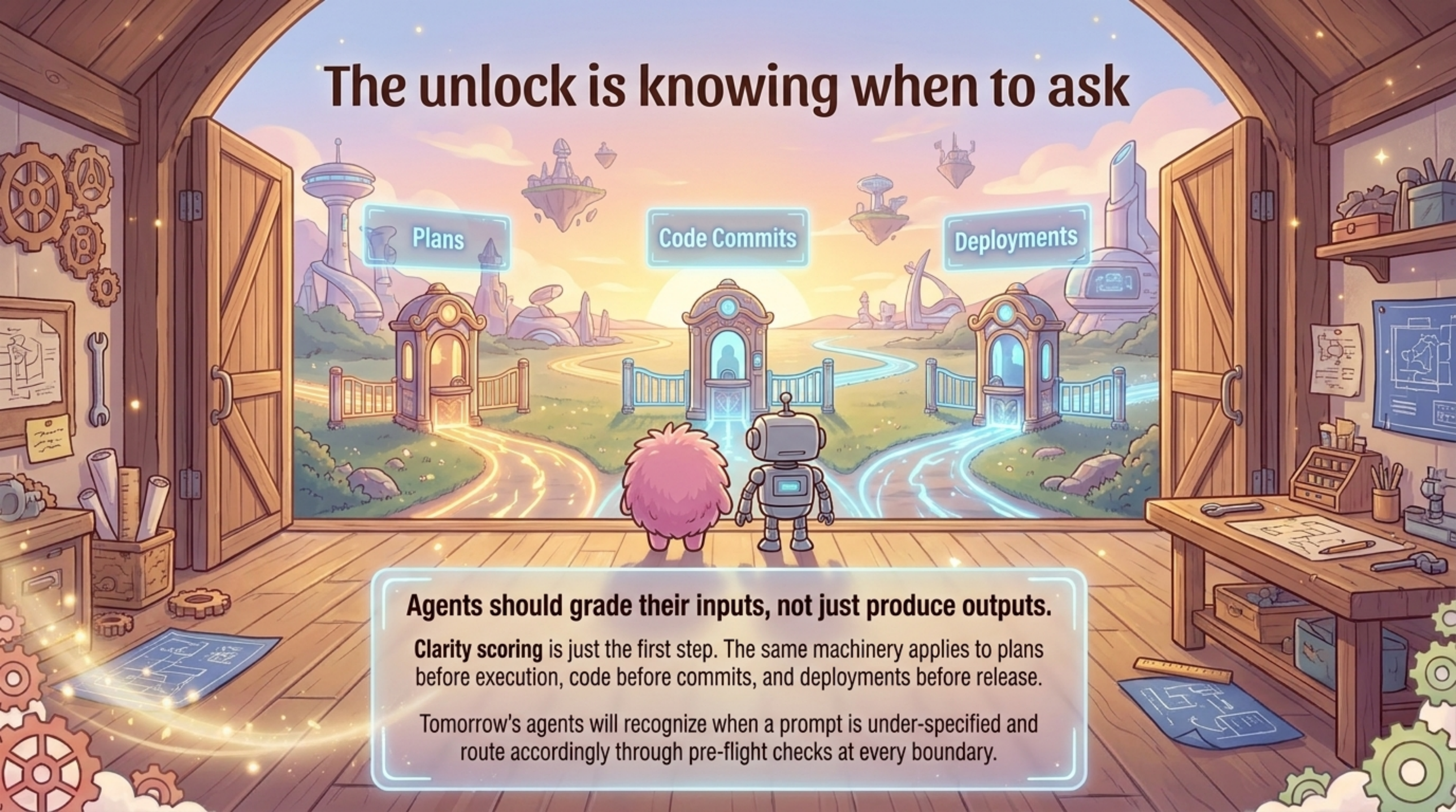
Build the minimum that produces a signal. Decide the roadmap based on that signal.

Open questions for the drafting board

Refining the heuristics of when to ask, when to infer, and when to step back.



The unlock is knowing when to ask



Agents should grade their inputs, not just produce outputs.

Clarity scoring is just the first step. The same machinery applies to plans before execution, code before commits, and deployments before release.

Tomorrow's agents will recognize when a prompt is under-specified and route accordingly through pre-flight checks at every boundary.