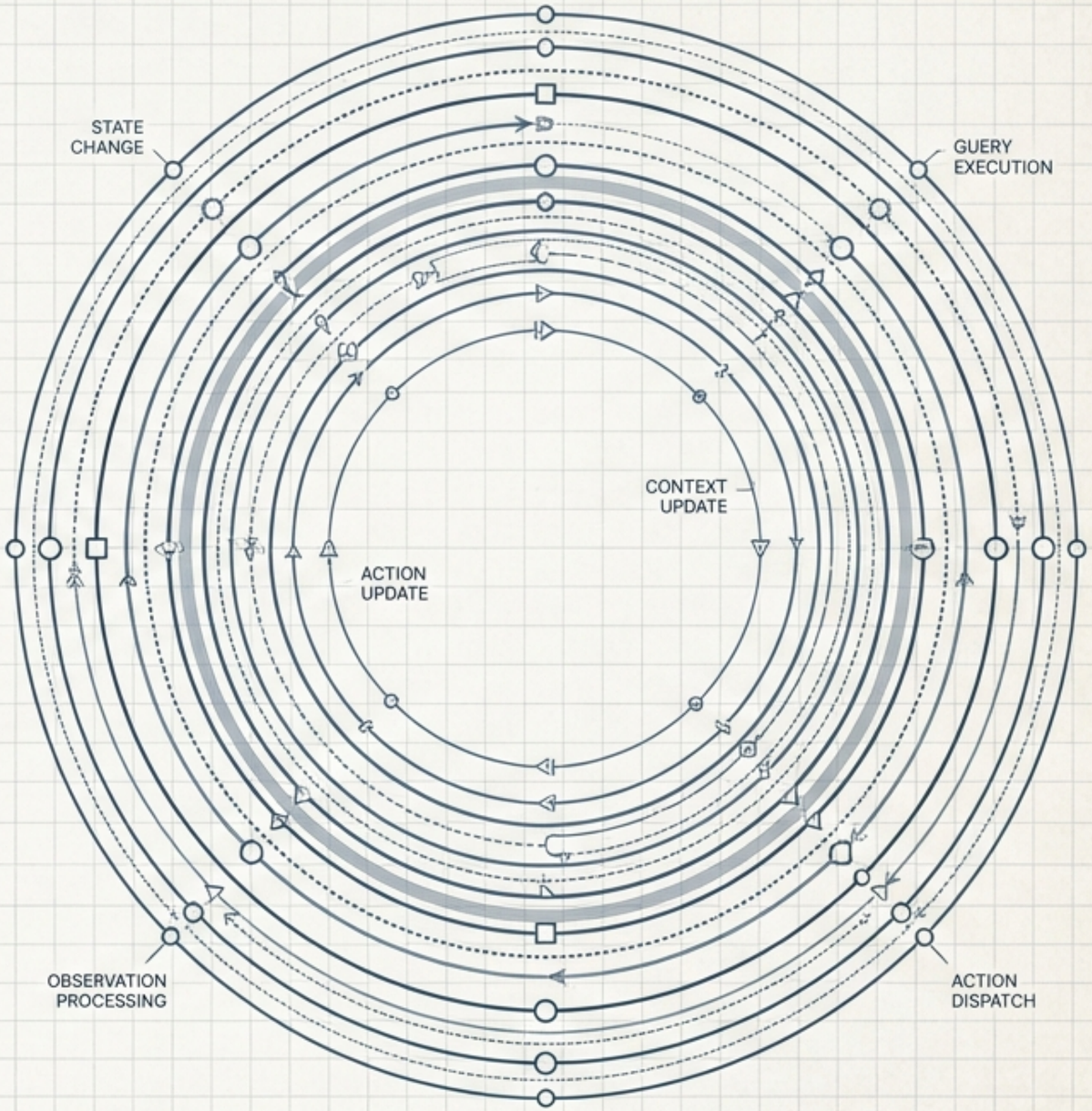
A Coding Agent Is Not a Prompt.

It’s a Stateful Query Loop.



Inside Agent Runtimes

The Popular Mental Model Is Too Simple

This linear, one-pass flow is fine for explaining the concept to a beginner.
It is terrible for explaining how real coding agents actually operate.

Smuggled Assumption:
Treating user requests as a single, uninterrupted pass through the model.

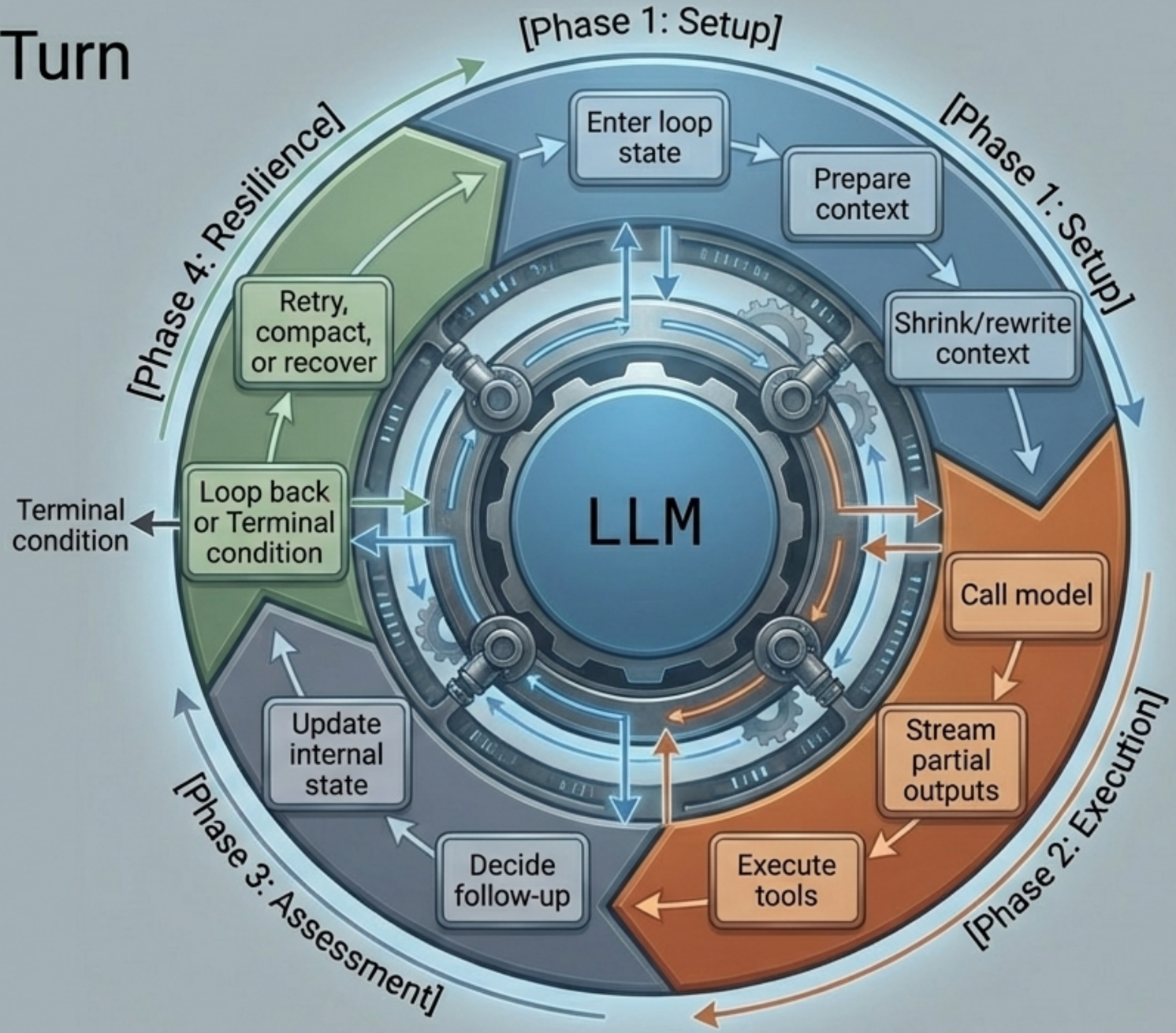
Two Mental Models of AI Agents

Dimension	The Naive Model	The Reality
Core Architecture	Stateless Prompt Wrapper	Stateful Query Loop
Execution Flow	One-pass	Multi-turn iteration
Error Handling	Surface immediately	Intercept and recover
Tool Integration	External to the loop	Integrated pipeline
Model’s Primary Job	Do everything	Choose actions & synthesize

An event loop with a language model inside it.

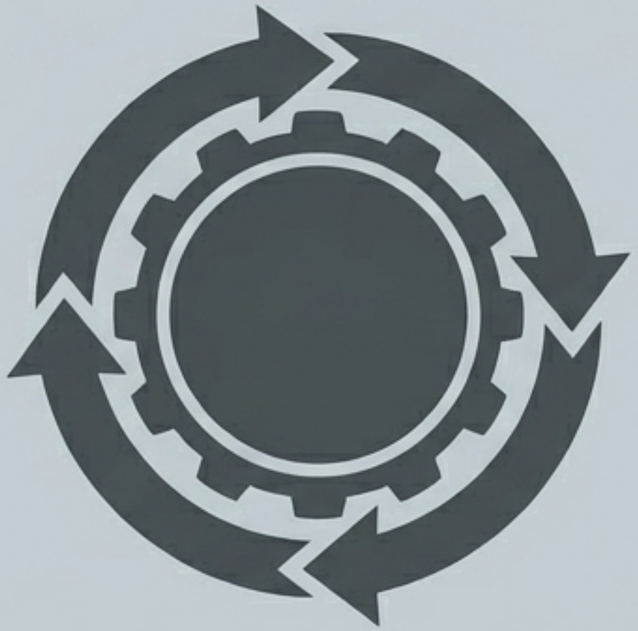
The best analogy for a top-tier coding agent isn't a “chatbot with tools.” The runtime operates a continuos cycle:

The Reality of a Single Turn



Agent behavior is strictly path-dependent.

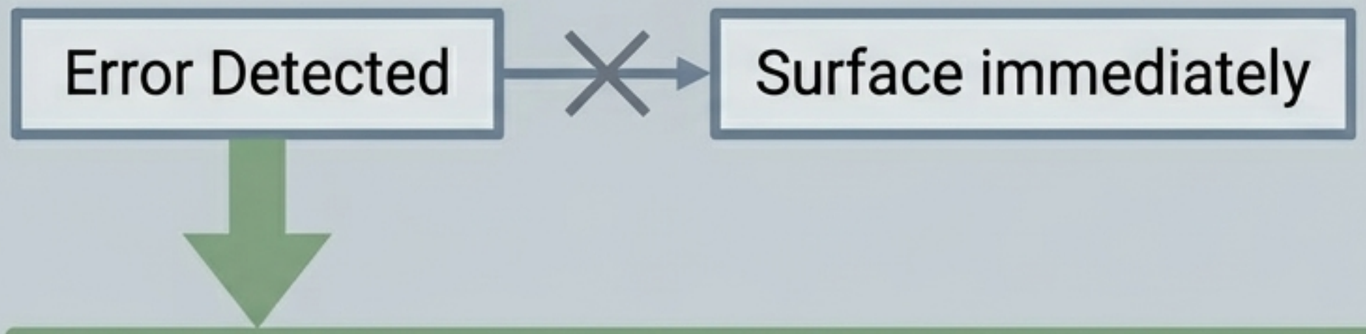
What the model should do next depends entirely on what just happened. You cannot express this if every step pretends it is a clean, stateless API call.



```
State Snapshot: {
    | messages:
    |   Array of active context,
    |
    | compaction_tracking:
    |   Status of context reduction,
    |
    | output_token_recovery_count:
    |   Integer tracking retry attempts,
    |
    | pending_tool_summaries:
    |   Results waiting for injection,
    |
    | stop_hook_state:
    |   Active interruption triggers,
    |
    | transition_reason:
    |  Logic driving the current loop turn
}
```

Engineers Who Do Not Trust the Happy Path

Recoverability is built into the heart of the main loop, not bolted on the side.



Recovery Subsystem

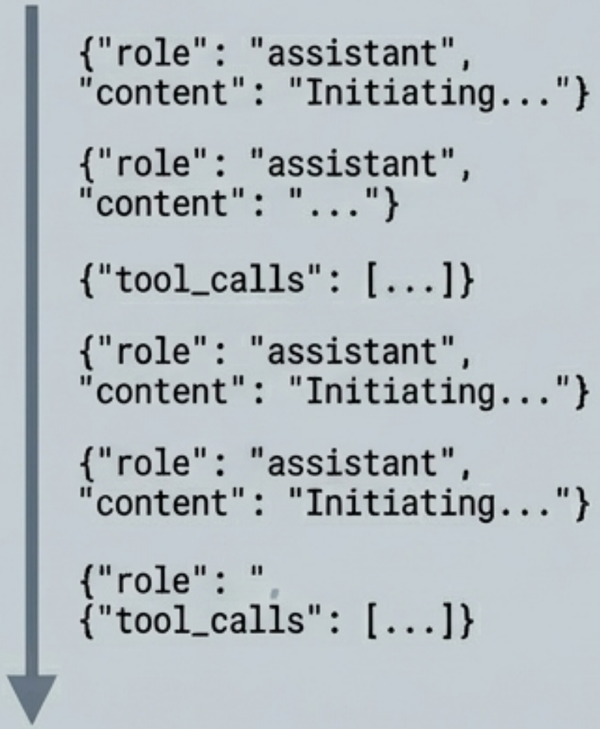
If the model returns a **recoverable** error (prompt-too-long, max-output-tokens), the system holds the error back. The runtime optimizes for "keep the agent alive" by triggering:

- Context collapse
- Reactive compaction
- Truncation retry

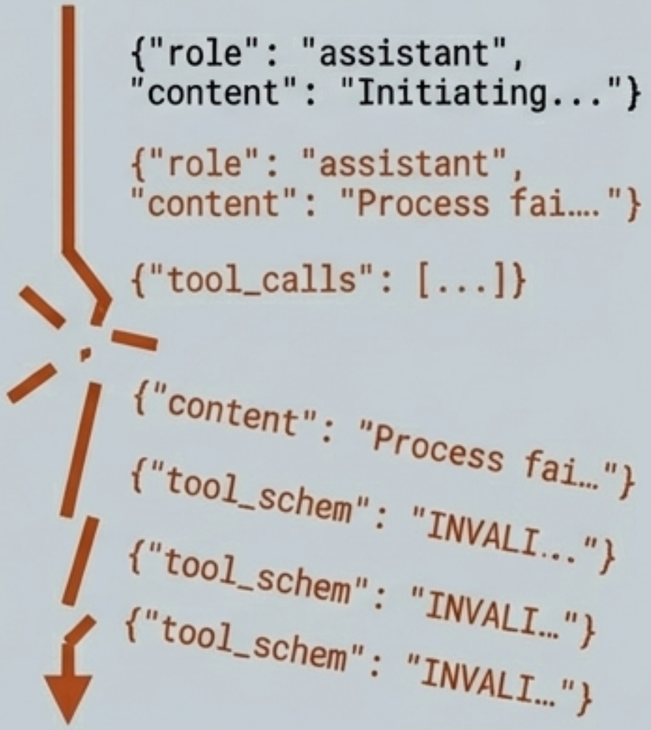
Tombstoning Partial Streams

A half-streamed assistant message is not harmless. It can poison the transcript, create invalid tool schemas, and break future API calls. If a streaming path falls back, the runtime explicitly emits tombstones to invalidate orphaned messages.

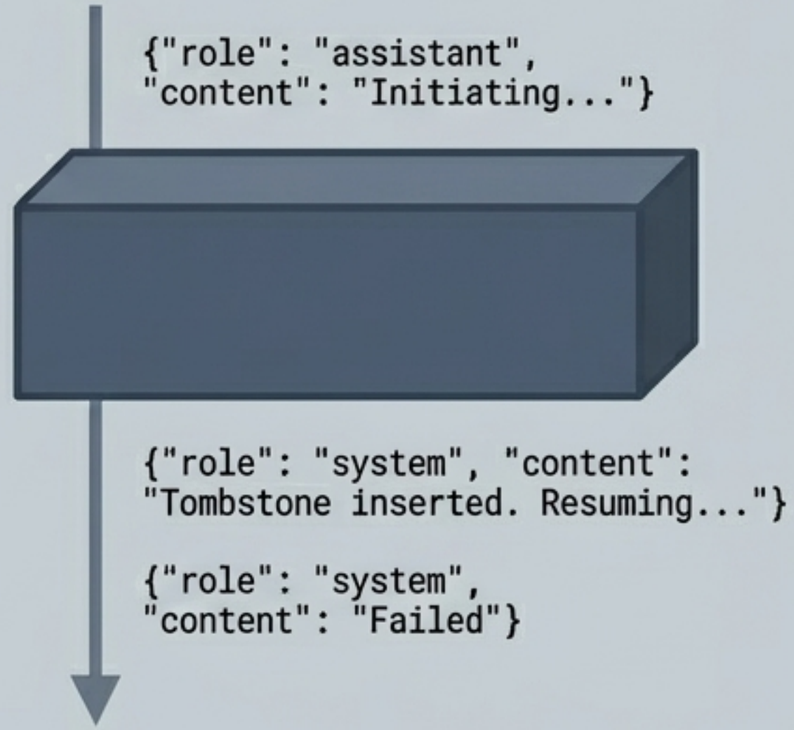
Before



During

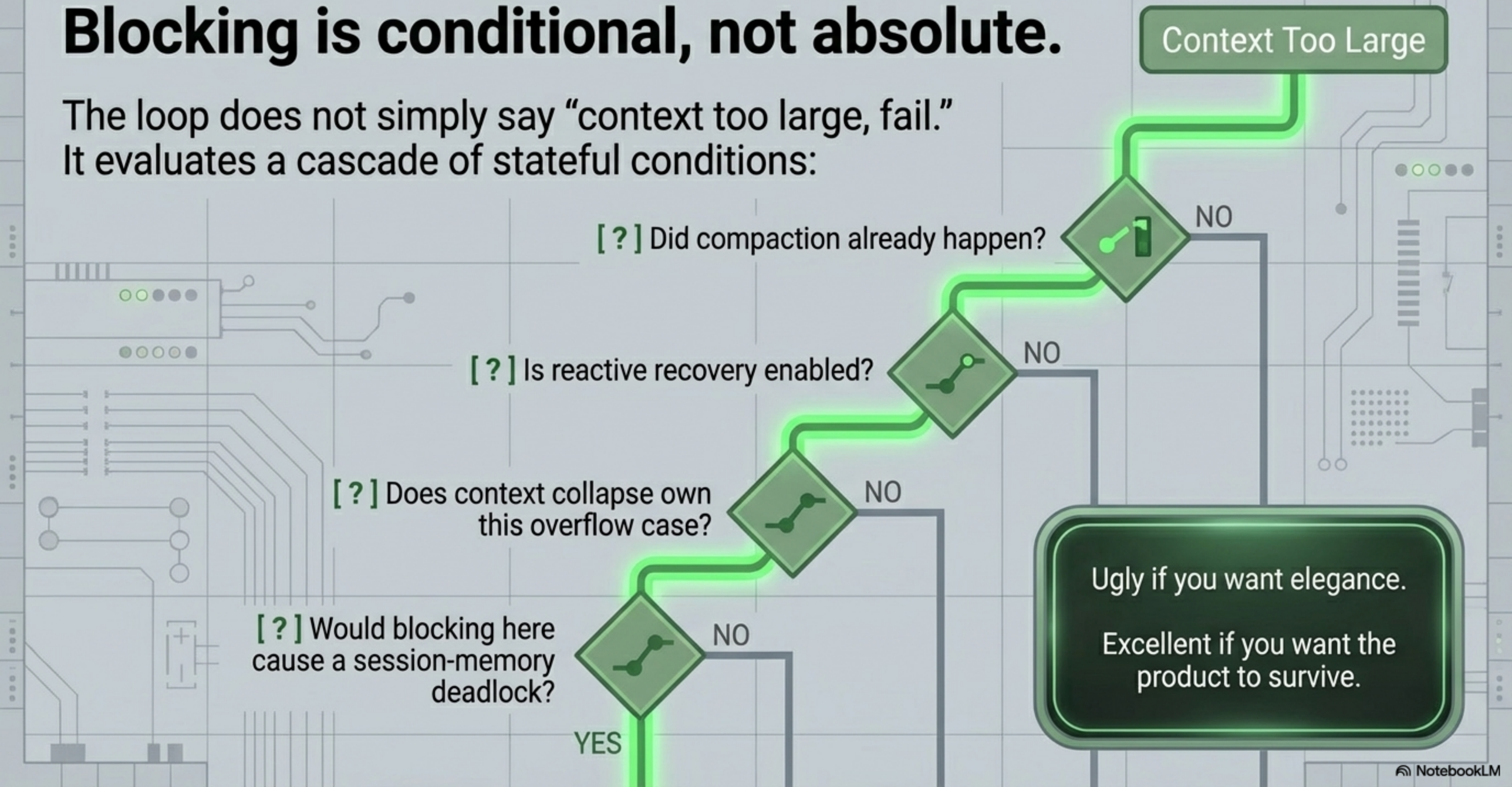


After



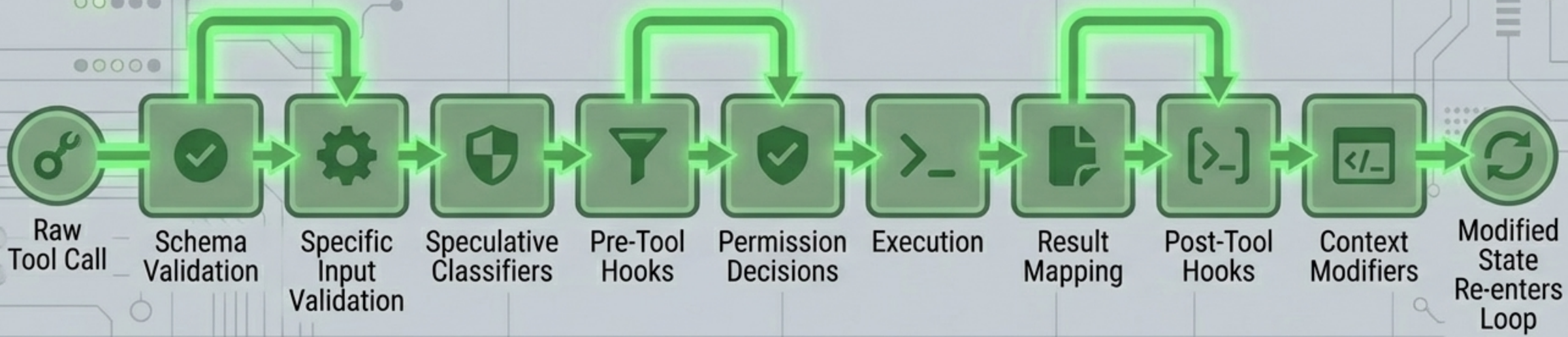
Blocking is conditional, not absolute.

The loop does not simply say “context too large, fail.”
It evaluates a cascade of stateful conditions:



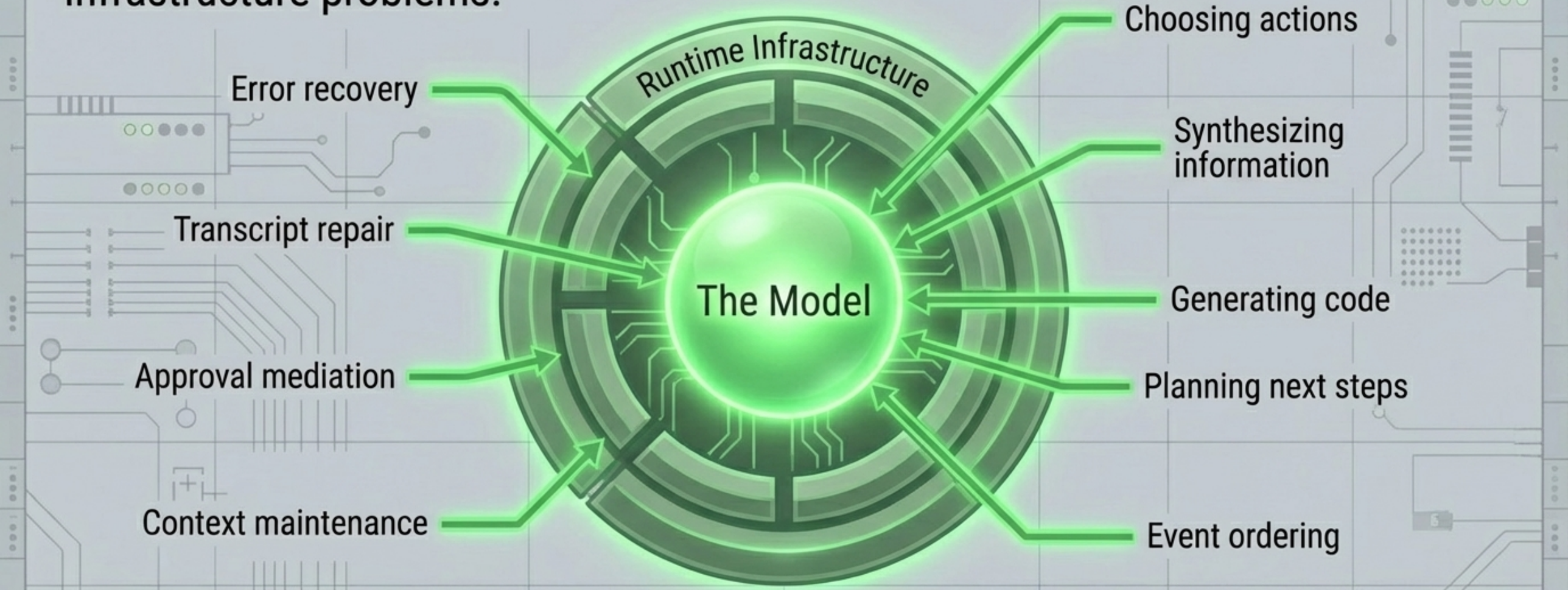
Tool execution is the loop, not a side effect.

The runtime doesn’t just hand calls to a helper and wait. It routes them through a heavily orchestrated policy and transformation pipeline.



The Model's Job Shrinks in a Good Way

Good architecture uses infrastructure to absorb infrastructure problems.



Do not ask:

What should my prompt say?

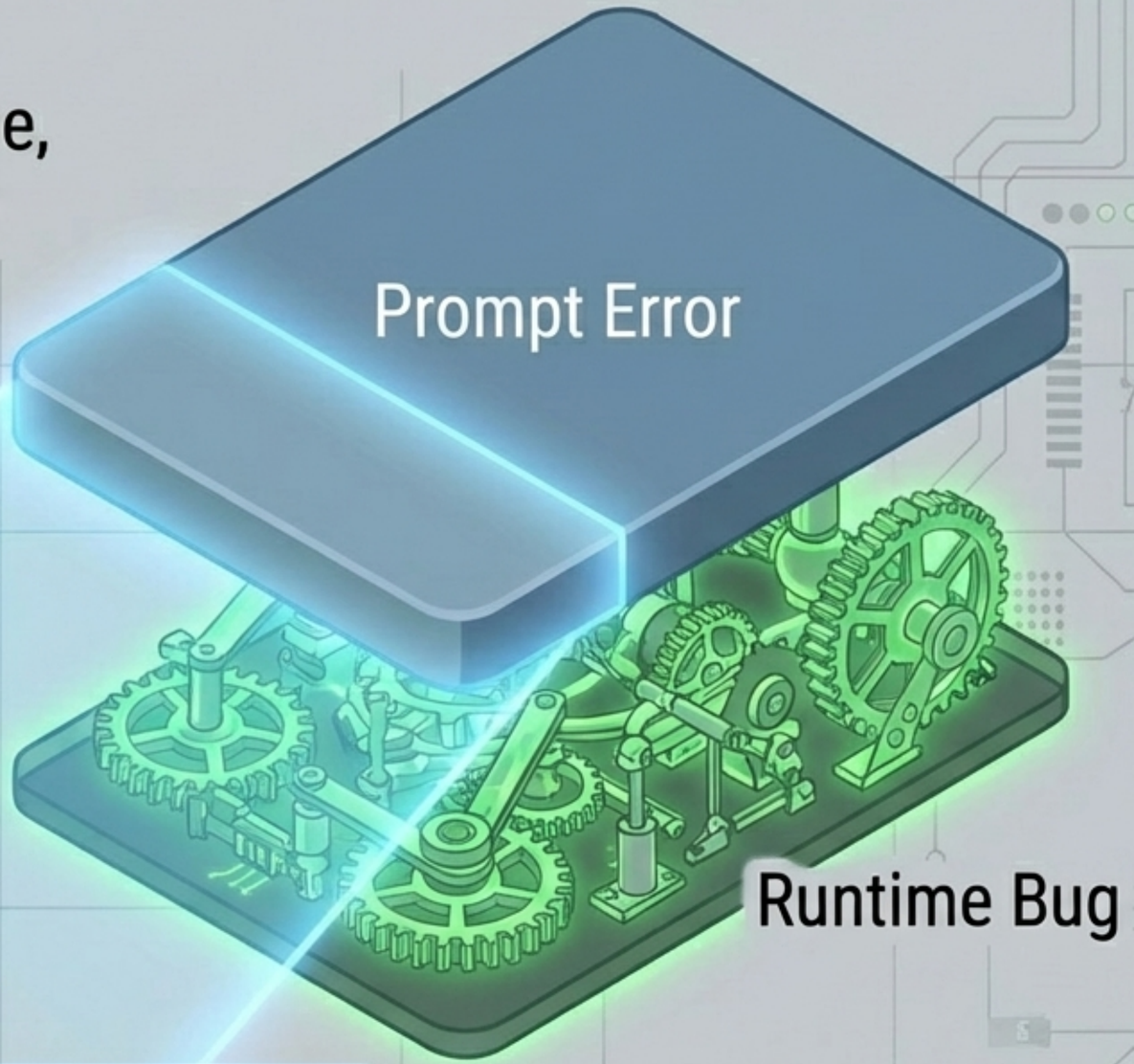
Start by asking:

What state do I need to carry between internal turns?

- [ ] Which tools are still in progress?
- [ ] Which errors were already retried?
- [ ] What parts of history have been summarized?
- [ ] What can still be trusted in the transcript?
- [ ] Which sidechain owns what state?

Look at the Loop.

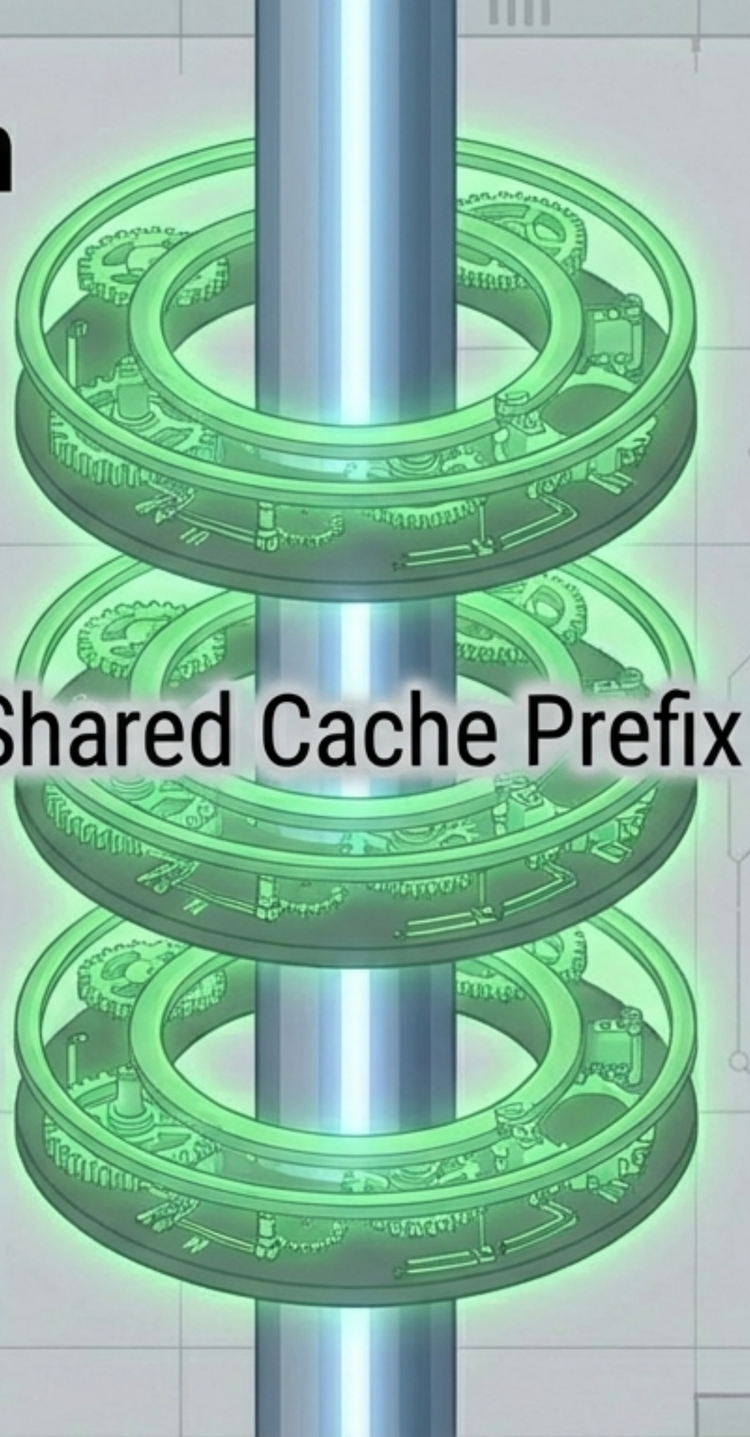
When your agent starts to feel unreliable, don’t only look at the prompt. Most real agent bugs are runtime bugs wearing prompt-shaped masks.



- Are failures classified well enough for recovery?
- Are partial outputs polluting the transcript?
- Is the right context transformation happening before the next model call?

The Next Uncomfortable Truth

Once you accept that an agent is a stateful query loop, performance and cost cease to be second-order details. They become structural.



Forked workers built around fake tool results so multiple children can share an identical cache prefix... **Prompt cache in serious agent systems is not an optimization. It is architecture.**