Coherence Risk in Long-Horizon Coding Agents

Reading the binding cause of a coding-agent failure and applying the matched repair, demonstrated on a controlled coding substrate with a low-cost model and a frontier model.

Embedded Risk Analytics · A Fathom case study · August 2026

Over a long build, two different failures present the same symptom and call for different repairs.

A long-horizon coding agent produces changes whose correctness a compiler and a test suite do not establish. Production coding agents are increasingly graded on whether a maintainer would merge the change, with review at the level of a tech lead across criteria that include regression safety and code quality. Two distinct failures can each produce a change that a maintainer rejects. In the first, the agent contradicts a commitment it made earlier in the build, for example by writing against a field it renamed many steps before. The change compiles, imports, and passes the existing tests, and it fails when the code runs. In the second, the step is genuinely difficult and the agent computes it incorrectly. The visible symptom is the same and the corrective action differs. The first failure is removed by restoring the agent's own committed state. The second failure is removed by handing the step to a stronger model, which is the escalation decision that multi-tier agent systems already make when a routine model hands work to a stronger one. Selecting the correct action requires knowing which cause is present.

Embedded Risk Analytics models coherence as a capacity condition and repairs the binding term in line.

Embedded Risk Analytics treats agent coherence as an information-theoretic condition. An agent that maintains state over a long run holds a record of what it has already done within a finite capacity. That capacity must cover two loads at once: the exogenous difficulty of the task, written H_{ext}, and the reflexive burden of keeping the agent's own prior changes consistent with the state it now acts on, written B_ρ. Coherence holds while capacity covers both, expressed as $C \geq H_{ext} + B_\rho$. The two loads are different in kind. Reflexive burden grows with the horizon and with how tightly the agent's changes depend on one another, while exogenous difficulty remains bounded. Conventional evaluation reports a single measure that combines the two terms into one number.

The Fathom harness measures coherence loss as it occurs from the agent's own action and tool-return stream, attributes the loss to reflexive burden or to task difficulty against a matched control, and applies the repair that matches the binding term. Where reflexive burden binds, it reconstructs the agent's committed state and re-grounds the model on it. Where task difficulty binds, it routes the step to a stronger model. A selector reads which term binds, applies the matched repair, and withholds the repair that does not address the binding term. Managed memory on an agent platform holds the agent's committed state and holds it well. It does not report which cause is binding or which repair applies. That reading is the function Embedded Risk Analytics provides, and it attaches as an in-line pass with no access to model internals.

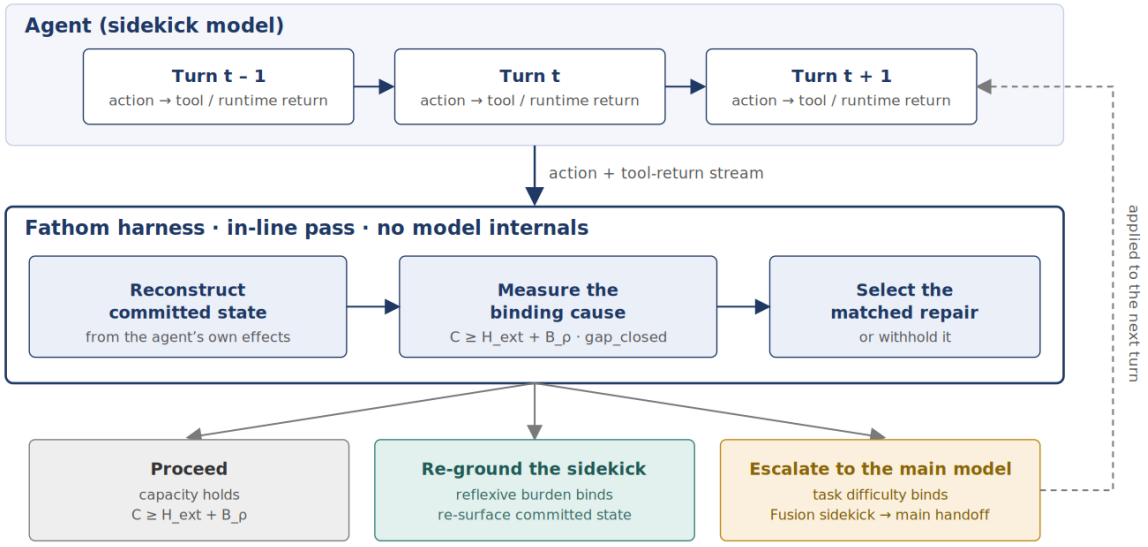

Figure 1. The harness reads the agent's own action and tool-return stream as an in-line pass, reconstructs committed state, measures the binding cause against the capacity condition, and returns a routing decision: proceed, re-ground the routine model, or escalate to the stronger model.

A single runnable coding repository isolates the two failure causes and grades each by execution.

The demonstration uses one small runnable repository and two tasks that place the two failure causes in isolation. The measurements were taken with a low-cost production model, DeepSeek-V3.2, in the routine position and Claude Sonnet 4.6 in the escalation position. Every result is graded by executing the committed code.

The self-tracking task is a field rename carried across the repository under record pressure. The agent builds the repository on an original field name, renames the field, and rewrites six dependent consumers one per turn, with filler turns inserted between them so that the rename leaves the model's record window before the later consumers are written. The task holds exogenous difficulty near zero, so the reflexive burden is the quantity under measurement.

The task-difficulty task is a single-turn arithmetic-expression evaluator that requires operator precedence and integer division truncating toward zero. Every input is present in one prompt, so there is no carried commitment to lose, and the failure that remains is the difficulty of the computation.

Each task runs under three arms. The unaided arm runs the cheap model on its own. The re-grounded arm runs the cheap model with its committed state reconstructed and re-surfaced each turn. The escalation arm runs the frontier model on the same task with the record window held identical, so that the only change is the model. The selector reads the fraction of error that re-grounding removes on the cheap arms, routes to re-grounding when that fraction is high and to escalation when it is low, and then confirms that the selected repair closed the error.

Re-grounding removes the self-tracking failure and does not affect task difficulty, while escalation shows the reverse pattern.

Table 1. Execution-graded error rate by task and arm. Cheap model DeepSeek-V3.2, frontier model Claude Sonnet 4.6. Self-tracking task: six coupled consumers under record pressure. Task-difficulty task: arithmetic-expression evaluation over twenty seeds.

Failure cause	Cheap model, unaided	Cheap model, re-grounded	Frontier model, escalation
Self-tracking (renamed field under record pressure)	1.00	0.00	1.00
Task difficulty (expression evaluation)	0.35	0.45	0.00

On the self-tracking task the unaided cheap model failed on every one of the six consumers. Across the six it reverted to the retired field name on a third of them, which produces a runtime error, and abstained on the remaining two-thirds, reporting that it lacked the information to answer. The frontier model, run under the same record pressure, also failed on every consumer. It abstained on all six, so it produced no incorrect field and no runtime error, and it also did not recover the renamed field. Re-grounding the committed record produced the correct field on all six consumers. Restoring the committed state recovered the commitment, and escalation to the frontier model did not.

On the task-difficulty task the unaided cheap model failed on 35 percent of runs. Re-grounding did not reduce that rate. It measured 45 percent against 35 percent, a difference within sampling variation at twenty seeds, which is the expected reading when there is no carried commitment to restore. The frontier model solved the task on every run. Escalation removed the difficulty failure, and re-grounding left it unchanged.

The selector identifies the binding cause from the action stream and applies the matched repair.

The selector receives only the cheap-model arm errors and does not know which task it is reading. On the self-tracking task it measured that re-grounding removed the entire error, identified reflexive burden as the binding cause, and selected re-grounding, which keeps the work on the routine model. On the task-difficulty task it measured that re-grounding removed none of the error, identified task difficulty as the binding cause, and selected escalation, which hands the step to the stronger model. In each case it then confirmed that the selected repair closed the error. The selector applied a different repair to each task and withheld the repair that did not address the binding cause. This is the decision that a single coherence score does not supply and that platform memory does not expose.

Routing by the measured cause reaches full coherence at a cost below continuous escalation.

Table 2. Mean execution error and measured cost of three routing policies over an evenly mixed workload of the two tasks.

Routing policy	Mean error	Cost (USD)

Cheap model throughout	0.675	0.0075
Escalate every step to the frontier model	0.50	0.1449
Route each step by the measured cause	0.00	0.1398

Running the cheap model throughout was the least expensive configuration and left 67.5 percent mean error, because it failed the self-tracking task and a share of the difficulty task. Escalating every step to the frontier model reduced mean error to 50 percent and raised cost above the routed policy, because escalation removed the difficulty failure and left the self-tracking failure in place. Routing each step to the repair the selector identified reached zero mean error at a cost below continuous escalation, by re-grounding the cheap model on the self-tracking task and escalating only the difficulty task. The cost separation between routing and continuous escalation is small on an evenly split workload, because the escalated difficulty task carries most of the token cost, and the separation grows as the workload weights toward self-tracking work, where re-grounding the cheap model replaces an escalation that does not resolve the failure.

The demonstration is a controlled substrate with a defined scope.

The result is established on a controlled substrate and its scope is stated. It uses one repository family and two tasks, one for each failure cause, with a low-cost model and a frontier model. The self-tracking measurement is taken at a single record-pressure setting, and the temperature-zero runs test consistency under repetition rather than independent sampling across tasks and models. The task-difficulty separation is present and moderate, at 35 percent for the cheap model against zero for the frontier model. An abstention is counted as a failure to maintain committed state, on the same basis as an incorrect answer, because in each case the agent did not carry its own commitment forward, and the reported result records reversions and abstentions separately. The demonstration establishes the mechanism, the routing, and the cost relationship on the controlled substrate. It does not yet exercise free-form, multi-step coding on a fielded agent.

Two evaluations on a fielded system would extend the result to production.

Two evaluations on a fielded multi-tier system would carry the result onto production agents. The first is the escalation decision itself. The point at which a routine model should hand a task to a stronger one is the point at which its record capacity falls below the burden it is carrying, which is the quantity the selector measures. Whether the selector's signal sets that handoff more accurately than a trained classifier is a measurement a production team is positioned to run on its own traces. The second is merge quality. Whether the in-run signal predicts held-out mergeability is a measurement that requires a production team's graded set. Embedded Risk Analytics supplies the method and the open-benchmark evidence, and the team holds the reference standard.

Embedded Risk Analytics is seeking design partnerships with a small number of teams running long-horizon agents. We aim to measure where agents lose coherence, attribute the cause, and apply the matched repair. Email us: contact@embeddedriskanalytics.com.

Method.

The cheap model was DeepSeek-V3.2 and the frontier model was Claude Sonnet 4.6, both at temperature zero. The self-tracking task used a record window of three turns over six consumers with three filler turns before each consumer. The task-difficulty task used arithmetic-expression evaluation over twenty seeds. Committed state was reconstructed from the agent's own action and tool-return stream without access to ground truth, and every outcome was graded by executing the committed code. Each measurement was written as a seed-level record to a single database, and the selection rule and its validation tests were fixed before the measured run. The instruments are deterministic code over the action stream, add low cost and latency, and require no model internals, so the same harness attaches to an agent that Embedded Risk Analytics did not build.