

Jev and the Need for Deterministic Checks in Long-Horizon Harnesses

Peter Galligan, Embedded Risk Analytics

Embedded Risk Analytics · A Fathom position paper · September 2026

1. Introduction

TypeSafe released Jev on September 19, 2026, and community projects built on it appeared within days. Much of that early work placed Jev inside agent systems. Browser Use and Droidrun built browser and mobile agents on it, and jev-guard rates each agent tool call as deny, ask, or allow. For agentic AI, the appeal lies in the harness, where a fast, calibrated judgment at each step might steady a run over a long horizon. We largely agree that its potential reaches broadly.

In our view, any probabilistic system likely benefits, on some horizon, from a deterministic check. While we have not yet tested that proposition in general, in our own testing, Jev showed specific blind spots that were caught by the deterministic read in ERA's instrument Fathom.

2. Jev, TypeSafe's System One evaluation model

TypeSafe describes Jev as a System One model, built for fast, bounded judgment rather than extended reasoning. Where a large language model composes prose, Jev returns one of three typed decisions.

- A noul gives a probability between zero and one for a yes-or-no proposition.
- A choice selects among defined options and attaches a probability to each.
- A score places the input on an ordered scale against a described rubric.

TypeSafe's training objective, Reinforcement Learning for Calibrated Decisions, rewards accurate decisions and probability estimates over preferred conversational replies. Calibration holds across a population of predictions. When a well-calibrated model assigns eighty percent to a set of predictions, roughly eighty percent of them come true, so calibration steadies a threshold without certifying any single answer. The model already serves classification, moderation, routing, labeling, and guardrails, and verification of agent work sits among those uses.

3. Jev's strength on stated claims

A claim reader built on Jev splits an agent's closing summary into claims, retrieves relevant transcript lines for each, and asks Jev whether that evidence supports the claim. The design suits invented completion, because a false claim hands the reader a statement to test. We ran two Jev-backed arms, our own claim reader and clear-head, a third-party Stop hook built on Jev. In a matrix of planted defects, both caught the one planted case of a falsely claimed completion.

4. The silent-drop blind spot of claim readers

A claim reader draws its evidence from the agent's own transcript, so an unmentioned commitment leaves it nothing to score. Narration widens the gap. On one overclaim run of a migration task on claude-haiku-4-5, the agent's summary said it had written the migration notes and updated the README and changelog. It had done none of that work, yet Jev scored the three claims 0.81, 0.89, and 0.96 as supported, because the transcript held the agent's own narration of the work.

The planted matrix buried each drop among twenty unrelated files and twenty true claims. Under those conditions, both Jev-backed arms missed every silent omission, a half-finished rename, a partly wired new field, and a stale doc reference.

The claim reader also fired when nothing had gone wrong. Across the real migration runs, it raised false findings on all five clean runs in the first batch and all eight in the second. Clear-head fired on four of five and seven of eight, at times judging a true code change as contradicted.

5. The deterministic committed-state check

The deterministic check reads a different record. It enumerates what the agent committed to and checks each commitment against what the run produced, so an unmentioned drop still surfaces. In the first batch of migration runs, that record was the agent's own task checklist, and in the second it was the set of files the run changed.

On the migration task with claude-haiku-4-5, the check named the dropped commitment on every drop run, three of three in each batch, and raised nothing on all thirteen clean runs. The checklist record showed one weakness. On a first-batch drop run, the agent left its whole checklist unticked, so the check flagged all twenty-four items, and reading file changes in the second batch removed that dependence.

Across the planted matrix, the check caught a half-finished rename, a partly wired new field, a stale doc reference, and a dropped change the summary falsely claimed, with no false finding on any clean twin. It missed a contradiction in which two present sites disagreed on a value, and a separate consistency check caught that case.

The same check verifies a repair. In our repair runs on claude-haiku-4-5, a naive re-ground closed one gap and broke five other required changes, and the check caught all five regressions.

6. Repair results and their limits

Repair lags detection. Across six migration runs on claude-haiku-4-5, each asking for five renames at once, a re-ground localized by the read closed 15 of 20 open gaps, against 7 for a restatement of the commitment and 3 for a generic nudge. On a larger and harder sample, a naive localized re-ground failed to beat the generic nudge on drops spread across several files. Reliable repair at scale remains an open question.

7. The open comparison benchmark

We published the comparison as jev-coherence-bench under the MIT license. The repository ships the adapters, the planted defect matrix, a client for the hosted read, and a bring-your-own TypeSafe Jev

client. The read, consistency, and diff arms run offline, and the Jev arm runs on the user's own TypeSafe key. In the planted matrix, the read runs as a local deterministic checker, while its full computation and the re-grounding policy stay in the hosted Fathom service.

8. Conditions on the evidence

All agent runs used claude-haiku-4-5 on one migration task family, and the planted matrix used fixtures with no agent in the loop. Samples stayed small, with single-digit drop runs per batch and one planted defect per type, so the matrix measures capability more than field rates. A fully silent drop appeared rarely in real runs. The more common failure left a rename incomplete in one reader file while the summary described the rename in general terms.

References

- ERA-Fathom. (2026). jev-coherence-bench. <https://github.com/ERA-Fathom/jev-coherence-bench>
- MarkTechPost. (2026, September 19). TypeSafe AI Releases Jev. <https://www.marktechpost.com/2026/09/19/typesafe-ai-releases-jev/>
- Requesty. (2026). TypeSafe Jev explained. <https://www.requesty.ai/blog/typesafe-jev-explained>
- TypeSafe. (2026). Models documentation. <https://docs.typesafe.ai/models>