



TURNING AI RISK INTO ADVANTAGE

AGENTIC AI · WHITEPAPER · SEPTEMBER 2026

Calibrated to What? Governing Jev, LLMs and Decision Models in Agentic AI

A probability tells you how strongly a model supports an answer. It does not tell you whether that answer is right for your population, whether the decision was ever validated, or whether the agent is allowed to act on it.

What a decision model changes about model risk, how to validate one per decision rather than per model, and where authorization has to sit instead.

Satyaki Ghosh Dastidar
AI Risk Consulting

At a glance

Decision models can make agentic AI easier to govern, but only if firms stop treating their probabilities as self-validating.

THE ARGUMENT IN ONE PARAGRAPH

A probability tells you how strongly a model supports an answer. It does not tell you whether that answer is right for your population, whether the decision was ever validated, or whether the agent is allowed to act on it. Decision models return bounded, typed answers that software can branch on directly, which removes a real failure mode and creates a subtler one: a confident, well-formed answer to a question nobody checked. The fix is not a better model. It is a deterministic control layer that decides when the model's intelligence is allowed to act.

0

independent calibration studies have been published on Jev

As of 24 September 2026

2

frontier models averaged to set the reference labels in TypeSafe's own evals

TypeSafe launch post, Sept 2026

0.82 → 0.60

confidence on the same ticket, described options versus bare labels

Spring AI write-up, Sept 2026

Five things to take away

- 1 **Calibration has to be asked “against what?”** TypeSafe's published workflow evidence measures agreement with the averaged predictions of two frontier LLMs. That is evidence of consensus-matching, not calibration against real outcomes in a bank's own population.
- 2 **Schema safety is not correctness.** Jev cannot return an answer outside its defined type. It can still return a confident, well-formed, wrong answer — and a pick-one question cannot say “none of these” unless you ask that separately.
- 3 **The question wording is part of the model.** Small changes to how options are described move the model's confidence materially. Decision contracts therefore need the same change control as model code.
- 4 **Governance must sit outside the model.** Authorization, domain coverage and action thresholds belong in a deterministic control layer. Model confidence should never substitute for authorization.
- 5 **Rigor should be proportionate.** A routing decision a human sees next does not need the validation depth of an autonomous, irreversible action. Scale thresholds and evidence with consequence, reversibility and autonomy.

WHAT TO DO FIRST

Register the decisions before you evaluate the model. A firm that asks “is Jev validated?” gets a vendor answer. A firm that asks “is this decision, with this wording, on this population, at this version, validated for this tier?” gets an answer it can defend.

1 The decision gap

Most of what enterprises need AI to do inside a workflow is not writing. It is deciding. Agentic systems make many small judgments per task, and each one determines what happens next: should this request be approved, or does it need a person? Which queue or workflow should handle it? Is this agent's proposed action within its authority? Does the evidence support the action being proposed? Is this behavior anomalous?

Large language models can answer these questions, often well. But they answer them the way they answer everything: by generating text. The software around them must then parse that text, validate it and hope the format holds. When the model is buried several layers deep in a dependency chain, a malformed or unexpected response is not an inconvenience. It is a failure mode.

There is a second problem. LLMs are not designed to say how sure they are. They can be prompted to state a confidence number, but that number is generated after the fact and tends to be overconfident. TypeSafe's framing of the problem is fair: models have been superhuman at chat for years, yet automation has lagged. Its explanation is that preference training made chat models overconfident, and overconfidence keeps a human in the loop.

Firms have historically filled this gap in two ways. Rules engines are deterministic and auditable but brittle when the condition is a judgment. Traditional machine learning produces well-understood probabilities but needs a stable feature vector, labeled outcomes and a development cycle per use case. Neither fits well when the input is a messy agent trace, a conversation, or a policy excerpt, and when the decision set keeps changing.

Decision models like Jev are aimed squarely at that middle ground: unstructured state in, a typed decision with a probability out, and ordinary code deciding what to do with it. Whether that middle ground can be governed is the subject of the rest of this paper.

2 What Jev is, and isn't

Jev is a text-in, typed-decision-out model: you send it state and a set of questions with predefined answer spaces, and it answers all of them in one parallel pass. TypeSafe describes a new model architecture, a parallel sampler and a training method it calls Reinforcement Learning for Calibrated Decisions (RLCD). It contrasts RLCD with the methods behind chat models: training on human preference ratings (RLHF) and on answers a program can verify (RLVR). The hosted model accepts text only; images or audio must be converted to text or structured observations first.

The three primitives

Every Jev question is one of three types. Their outputs differ more than many early write-ups suggest, and the differences matter for governance.

EXHIBIT 1 The three question types and what each returns

Primitive	What it asks	What it returns	Confidence field?
Noul	A yes/no question	A single number from 0 to 1: the probability of “yes”	No; the probability is the certainty
Choice	Pick one option from a defined set (up to 255)	The selected option plus a probability for every option	Yes
Score	Place the input on an ordered rubric (2 to 10 levels)	A continuous value along the rubric plus a probability per level	Yes

Source: TypeSafe documentation, accessed 24 September 2026.

Two details deserve emphasis.

Noul is not a two-option Choice. It returns one probability, not a “YES 0.91 / NO 0.09” pair with a separate confidence. What “yes” means depends entirely on how the question is worded.

A Score value is not a rounded level. On a three-level rubric scored 0 to 2, Jev can return 1.1, a point just past the middle level. That makes thresholds on Score meaningful, but it also means the number is a position on a rubric the firm wrote, not a measured quantity.

What Jev is not

It is **not a generator**: it cannot write a customer letter, an explanation or code, and anything that must be language needs a different model. It is **not a calculator** either — arithmetic and date logic belong in code, and TypeSafe lists both among Jev’s known weak spots.

3 Claims ledger

TypeSafe’s launch material is unusually candid about its own caveats, which makes it possible to sort its claims into three groups: easy to check, structurally true but narrower than they sound, and not yet independently tested. A governance team should know which is which before a vendor conversation, not after.

EXHIBIT 2 What is checkable, what is self-graded, and what is untested

Claim	What TypeSafe says	Status	What a governance team should note
Latency	70–500 ms end to end	Checkable	Evals were run from laptops on the US West Coast. Measure from your own network and concurrency.
Cost	\$0.042 per million input tokens; output free	Checkable today	TypeSafe says it cannot prove the price is not subsidized. Treat long-run pricing as a vendor risk.
Speed and cost vs. LLMs	20–200x faster, 40–400x cheaper	Self-graded	Measured on TypeSafe’s own workflows. Your workload will differ.
No type errors	Mathematically impossible	True by construction	Holds for schema conformance. Says nothing about whether the answer is correct.
“Can’t hallucinate”	Stated in the launch post	Narrower than it sounds	A schema guarantee only. TypeSafe’s own FAQ acknowledges Jev can still get things wrong.
Similar intelligence to LLMs on System One tasks	Based on TypeSafe’s workflow evals	Self-graded	Reference labels are the average of two frontier LLMs (GPT-6 Astra and Fable 5.1). Workflows were written by TypeSafe’s own capabilities team.
Calibrated probabilities	Core design goal of RLCD	Not independently tested	No outside party has published a calibration study.
Architecture and training method	New architecture; RLCD	Undisclosed	No public weights, no paper on RLCD and no architecture description.
Consistent answers for similar inputs	Stated in the launch post	Unverified	At least one gateway provider reports answers can vary between calls. Reproducibility must be measured.
Every decision has a confidence estimate	TypeSafe homepage	Imprecise	Only Choice and Score expose a confidence field, and it is derived from the probabilities rather than a separate output. Noul returns a probability only.

Sources: TypeSafe launch post, homepage and documentation; *The New Stack*; You.com explainer; Pere Pages analysis; Opper gateway documentation; TrueFoundry review. All accessed 24 September 2026.

The claims easiest to verify — speed, cost, type safety — matter least for governance. The one that matters most, calibration, has the least external evidence, and without a paper or architecture description it cannot yet be examined from the outside.

The benchmark design deserves credit and scrutiny in equal measure. Fixing the workflow and comparing every model against it is a sensible way to test a component. But a frontier-model consensus reference measures *agreement with other models*, not correctness against observed outcomes. TypeSafe does publish its own list of weak spots for the current version (Jev 1.13): arithmetic, dates, distractors, adversarial state and structural inconsistencies in the input. That list is a ready-made starting point for robustness testing.

4 Probability, confidence, calibration

These three terms are often used interchangeably, but they answer different questions. The first two are properties of the model’s output. Calibration is a property of how that output relates to the world: a vendor can train for it, but only the deploying firm can demonstrate it.

EXHIBIT 3 Three words that are not synonyms

Term	The question it answers	Who can establish it
Probability	How strongly does the model support this answer relative to the alternatives?	The model, at inference time
Confidence	How concentrated is the model's distribution for this Choice or Score?	The model, at inference time
Calibration	When the model says 0.8, does the event happen about 80% of the time in <i>our</i> population?	Only the firm, against observed outcomes

The third row is the one that carries supervisory weight, and the only one a vendor cannot supply.

Calibrated to what?

This is the question most commentary on Jev skips. RLCD is described as training for epistemically honest probabilities. But honest relative to what reference? TypeSafe's published workflow evaluations use the averaged predictions of two frontier LLMs as the reference probabilities.

THE DISTINCTION THAT MATTERS

If that same kind of reference shapes training, the most that the public evidence can support is this: *Jev's probabilities track what frontier models would say*. That is useful. It is also a different claim from *Jev's probabilities track what actually happens*. Frontier models share training data, blind spots and biases. A model calibrated to their consensus can be confidently wrong in exactly the places they are all wrong.

For a bank, the distinction is concrete. A fraud, complaint-severity or policy-breach decision has an observable outcome: the transaction was fraud or it was not, the complaint was upheld or it was not. Calibration has to be measured against that, on the firm's own population, before any threshold is set.

Calibration is local

Even a well-calibrated model is calibrated somewhere. A model can be well calibrated on consumer card transactions and poorly calibrated on commercial wires. It can be calibrated this quarter and drift next quarter as the population or the agent's behavior changes. A single global Jev model does not imply globally valid thresholds.

Three practical traps

Rounding at the tails. TypeSafe rounds probabilities and scores to two decimal places. A "0.99" therefore covers a band of true values, and a "1.00" is not certainty. Thresholds at the extremes, which is exactly where automation thresholds tend to sit, are coarser than they look. Validation must also show there is enough data in those tail buckets to say anything at all.

Noul has no confidence field. A Noul probability of 0.55 and one of 0.99 both arrive as a single number. Governance logic that expects a separate confidence value for every decision will silently misbehave on Noul questions.

Calibration does not compose. Well-designed Jev workflows ask several narrow questions about the same state and combine the answers in code. Each question may be individually calibrated, yet

the combined decision need not be. The questions share an input and are therefore correlated. A rule such as “escalate if unauthorized > 0.9 or harm > 0.8” has its own error rate, which must be measured directly rather than inferred from its parts.

5 The unknown-state problem

The hardest governance risk with Jev is not a wrong answer on a validated decision. It is a confident answer on a decision nobody validated. Goal-directed agents create paths their designers did not enumerate. When a decision model sits across all of those paths, some of its answers will land in territory where its probabilities have never been checked.

It helps to separate three kinds of uncertainty, because each needs a different control.

EXHIBIT 4 Three uncertainties, three different controls

Uncertainty	The question	Who can answer it
Model uncertainty	Is Jev unsure about this answer?	Jev, through its probabilities
Domain uncertainty	Have we shown Jev’s probabilities mean anything for this kind of decision and population?	The firm’s validation evidence
Governance uncertainty	Is the agent allowed to take this action at all?	Policy and entitlements, deterministically

A high Jev probability addresses only the first row. A 0.97 on an unvalidated decision tells you nothing about the second, and nothing at all about the third.

Forced choice makes this worse

Type safety is usually presented as pure upside: a Choice question cannot return an option outside its set. The flip side is that it cannot decline either. A Choice always names a winner. Practitioners are already advising that “none of these” be asked as a separate Noul question.

This matters for the unknown-state problem. An out-of-scope case does not produce an error. It produces a well-formed, in-scope answer, often with a respectable probability. Nothing in the output flags that the question did not really apply.

The coverage gate is circular

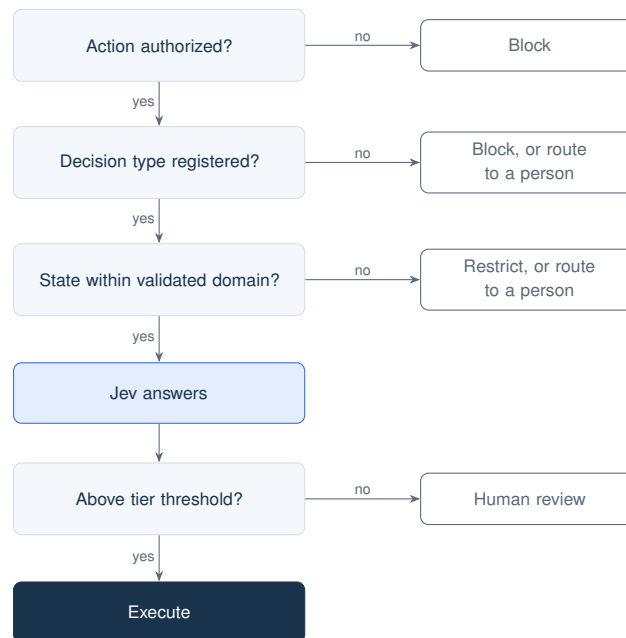
The natural control is a coverage gate: before a Jev answer can drive a consequential action, check that the decision is registered and the state is within the validated domain. The difficulty is that “is this state within the validated domain?” is itself a judgment. The obvious tool for that judgment is another Jev question, which would ask the model to certify its own applicability.

That circularity cannot be fully engineered away, but it can be reduced by drawing coverage evidence from signals that do not come from the same model:

- **Registration, deterministically.** Every decision Jev may influence has an ID, an owner and an approved action set. Unregistered decision types fail closed.

- **Structural checks on the state.** Required fields present, schema version known, source systems on an allowlist, input length inside the validated range.
- **An explicit out-of-scope question.** A Noul asking whether the case fits the defined options. It is not independent of Jev, but it is at least a separate, testable question.
- **Independent novelty signals.** Distance from the validation population using a separate method, such as a different embedding model or simple feature statistics.
- **Novel-state monitoring.** The rate of low-coverage cases tracked over time, since a rising rate is itself a sign that the agent's behavior has drifted.

EXHIBIT 5 The order of the gates is the control



The order is deliberate. Authority is checked first and deterministically, so Jev is never consulted on an action the agent may not take. Jev is consulted only after registration and coverage checks establish that its answer can be relevant, and the decision's tier sets the threshold it must clear.

6 Intelligence layer vs. control layer

The defensible way to deploy Jev is as an intelligence layer that feeds a separate, deterministic control layer. The model characterizes the situation; code and policy decide what is permitted. Confusing the two is how model confidence ends up substituting for authorization.

EXHIBIT 6 Each layer has exactly one job

Layer	Job	Nature
LLM	Reason, plan, draft, converse	Flexible, generative
Jev	Characterize the proposed action through narrow questions	Probabilistic, typed
Control layer	Apply authority, coverage, thresholds and risk tier	Deterministic, auditable
Human	Judge where automation is not yet trustworthy	Accountable
Execution	Carry out only what was permitted	Constrained by entitlements

The control layer is the only one that decides. Everything above it describes.

Decompose, then combine in code

TypeSafe’s own guidance is to break a judgment into several narrow questions and combine them with your own formula, rather than asking one broad question. This pattern happens to suit governance well, and its value does not depend on Jev: in TypeSafe’s own comparison, decomposition made every model better, faster and cheaper before Jev was added. For an agent proposing a refund, the questions might be:

- Is the amount within the agent’s delegated authority? (A rule, not a Jev question.)
- Would an error materially harm the customer? (Noul)
- How reversible is the action? (Score)
- Which policy category does the request fall under? (Choice)
- Does the case fit any of the defined categories? (Noul)

The first item is the point: anything that *can* be computed deterministically should be. Jev is for the parts that are genuinely judgments.

Why this separation helps audit

Because the final action comes from code, every decision leaves an auditable record: the state supplied, the exact question wording and version, each probability returned, the threshold applied, the rule that fired and the action taken. That is far easier to examine than a paragraph of LLM reasoning.

A CAUTION ON EXPLANATIONS

A common suggestion is to let Jev decide and then ask an LLM to explain the decision. In a regulated context, that is a rationalization, not an explanation. The LLM did not make the decision and has no access to why Jev scored as it did. Where an explanation carries legal weight, such as a reason for declining credit, it must derive from the actual decision logic: the narrow questions asked, their answers and the rule that combined them. The decomposed design makes that possible. A post-hoc narrative does not.

7 Proportionate treatment

Not every Jev decision deserves the same rigor, and treating them all alike wastes effort on trivial decisions while under-controlling dangerous ones. Validation depth, threshold strictness and human involvement should scale with three properties of the *decision*, not of the model:

- **Consequence:** what an error costs a customer, the firm or a third party.
- **Reversibility:** whether a wrong action can be caught and undone before harm lands.
- **Autonomy:** whether a person sees the output before anything happens, or the output triggers action directly.

The same Jev model can serve decisions at every level. The tier belongs to the decision contract.

EXHIBIT 7 An illustrative tiering

Tier	Typical decision	Human role	Threshold posture	Evidence expected before use
Advisory	Suggest a queue or tag for a person to confirm	Human decides every case	Loose; probability shown, not enforced	Spot-check accuracy; basic monitoring
Routing	Route tickets or cases automatically, with easy correction	Human reviews samples and exceptions	Moderate; low-probability cases go to a person	Accuracy and calibration on a representative sample
Conditional automation	Approve or block a bounded action within set limits	Human handles everything below threshold	Strict; set from error costs on validated data	Full calibration study by segment; workflow-level testing; drift monitoring
Consequential action	Inform a consequential, hard-to-reverse action	Human approves before execution; Jev informs, never authorizes	Default deny; fail closed on any gap	Everything above, plus stress and adversarial testing, and independent review

The tier names are illustrative. What matters is that a firm fixes the tier when it registers the decision, and that the tier then dictates both the controls and the evidence required.

Thresholds come from error costs, and only work if calibrated

For a yes/no decision with known costs, the classical rule is to act when the probability of “yes” exceeds a cost ratio:

$$p \geq \frac{C_{\text{false act}}}{C_{\text{false act}} + C_{\text{missed act}}}$$

The top term is the cost of a wrong approval: acting when you should not have. The bottom adds the cost of a missed one: failing to act when you should have. If a wrong approval costs nine times a missed one, the threshold sits at 0.9.

The rule has a hidden precondition: it is only valid if the probability is calibrated on the population where it is applied. Apply it to an uncalibrated probability and the threshold still produces a number, but the error rates it was meant to deliver will not appear. This is why calibration evidence has to come before threshold-setting, and why the tier determines how much calibration evidence is enough.

Thresholds belong to the firm’s decision policy, not to the vendor. A vendor-suggested threshold is a default, not a control.

8 Validating a Jev decision

The unit of validation is the decision contract, not the model. A firm that asks “is Jev validated?” will get a vendor answer. A firm that asks “is this decision, with this wording, on this population, at this version, validated for this tier?” will get an answer it can defend.

The decision contract

Each registered decision should record: its ID and owner; the state schema supplied; the exact question and option wording; the primitive type; the intended population; the tier; the threshold and the action attached to each range; the fallback when coverage fails; the pinned model version; and the validation evidence behind all of it.

EXHIBIT 8 Four levels of evidence

Level	The question	What to measure
1. Capability	Does Jev get this decision right?	Accuracy, precision and recall, confusion matrix, against a defensible reference outcome and against the current baseline (rules, ML, LLM or human)
2. Probability quality	Does 0.8 mean about 80% here?	Reliability curves, Brier score, log loss, calibration by segment and by probability bucket, including the tails
3. Decision performance	Does using it improve the business decision?	Error costs at the chosen threshold, escalation rate, reviewer workload, customer impact, latency, cost per decision
4. Workflow behavior	Does the whole agent behave correctly with Jev embedded?	End-to-end outcomes over multi-step runs, error propagation, behavior on paths nobody designed

Level 4 is where most programs will be weakest and where the risk actually sits. Individually good decisions can still add up to a poorly controlled workflow.

The reference outcome deserves as much scrutiny as the model. Historical outcomes, independent adjudication and expert labels each carry their own biases. Frontier-model consensus, used in TypeSafe’s own evals, should be treated as a proxy at best.

The wording is a model parameter

Jev’s answers depend heavily on how the question and options are written. In one published test, describing the options properly produced a confidence of 0.82; bare labels on the same ticket dropped it to 0.60.

The governance consequence is direct. A developer who “tidies up” an option description has changed the model’s behavior as surely as retraining it. Question and option text should be version-controlled, change-approved and re-validated at the tier’s required depth.

Pin the version

Integrations expose both pinned versions (for example Jev 1.13) and a moving *jev-latest* alias. A validated threshold is meaningless if the model behind it can change without notice. Anything above the advisory tier should call a pinned version, and a version change should trigger re-validation.

Test reproducibility and robustness explicitly

Answers can vary between calls. Measure how often the same input produces a different branch at your threshold, because an audit that cannot reproduce a decision is a weak audit.

For robustness, start from TypeSafe's own list of weak spots: arithmetic, dates, distractors, adversarial state and structural inconsistencies. Add the stresses agents create: long traces, missing fields, contradictory evidence, tool output that changes the state mid-run, and populations that shift over time.

Treat it as a third-party model

Jev is a vendor model under any reasonable model-risk definition. There are no public weights, no paper on RLCD and no architecture description, the service is hosted, and the product entered early access only in September 2026. Standard third-party questions apply: data residency and retention, service continuity, pricing durability, change notification, and an exit path if the vendor changes direction.

Monitor continuously

Track calibration drift, population drift, escalation and override rates, novel-state rate, branch-flip rate on replays, and model-version changes. The monitoring cadence should follow the tier.

9 Right tool for the decision

The useful question is not whether Jev is smarter than an LLM or a traditional model. It is which architecture fits the decision in front of you. Most mature deployments will use all three, plus rules.

EXHIBIT 9 Four architectures, four different jobs

	Rules	Traditional ML	LLM	Jev
Best when	The condition can be computed exactly	The target is well defined, labeled history is plentiful, features are stable	The output must be language, code or open-ended reasoning	The input is unstructured, the answers are known in advance, volume or latency matters
Input	Structured fields	Engineered feature vector	Text and conversation	Text or structured state
Output	Deterministic value	Probability or score	Free text	Typed answer plus probabilities
Uncertainty	None; it is exact or it is wrong	Well understood, measurable	Weak; self-reported confidence tends to be overconfident	Native, but calibration unproven externally
Validation maturity	High	High; established practice	Emerging	Early; little independent evidence
Typical examples	Authority limits, eligibility cut-offs, arithmetic	Probability of default, transaction fraud scoring	Customer replies, summaries, drafting explanations	Triage, routing, escalation, action approval, checking LLM outputs

Most institutions will run all four. The failure mode is using one where another was already working.

Three judgments follow.

Keep validated ML where it works. A probability-of-default model built on stable financial variables, with years of validation history and established supervisory expectations, should not be displaced by a general decision model without evidence that the switch improves outcomes.

Use Jev where feature engineering breaks down. Deciding whether an agent's proposed action needs approval, from a conversation, a tool trace and a policy excerpt, is hard to turn into a stable feature vector. That is Jev's natural territory.

Pair Jev with LLMs rather than choosing between them. The LLM plans and drafts; Jev checks the proposal against narrow questions; the control layer decides. TypeSafe itself lists verifying LLM prompts, reasoning traces and outputs as a core use case. Used this way, Jev functions as a fast, cheap guardrail around a slower, more flexible model.

10 Open questions and conclusion

Jev is a credible and genuinely different piece of engineering, and it points in a direction governance teams should welcome: AI that returns bounded answers software can control, rather than prose software must interpret. But several questions remain open, and a firm considering production use should hold them explicitly.

- **Outcome calibration.** Do Jev's probabilities hold up against observed outcomes, not model consensus, on a regulated population? No independent study has answered this yet.

- **Calibration under agentic drift.** Can useful calibration be maintained across heterogeneous, dynamically generated states as agents evolve? This is materially harder than performing well on a fixed workflow.
- **Coverage without circularity.** Which independent signals reliably tell a firm that a state is outside the validated domain?
- **Composed error rates.** How should firms validate decisions built from several correlated Jev answers combined in code?
- **Reproducibility for audit.** How stable are branches at operational thresholds across repeated calls and version changes?
- **Vendor durability.** Will pricing, hosting and version policy remain stable enough to support a validated control?

Conclusion

Decision models change where model risk lives rather than removing it. The risk moves out of parsing and prose and into three places: the wording of each decision contract, the meaning of each probability on the firm's own population, and the boundary between what has been validated and what an agent can reach.

The response is architectural and proportionate. Register every decision Jev may influence. Tier it by consequence, reversibility and autonomy. Calibrate it against outcomes rather than consensus. Pin its version and control its wording. Wrap it in deterministic checks for authority and coverage, and send whatever falls outside to a person.

THE GOVERNING PRINCIPLE

Use AI to provide intelligence, and use governance engineering to decide when that intelligence is allowed to act. Model confidence should never substitute for authorization.

Sources

All pages accessed 24 September 2026.

- *Introducing System One Models & Jev*, TypeSafe AI launch post, 15 September 2026.
- TypeSafe AI homepage.
- TypeSafe documentation: Introduction.
- *TypeSafe launched Jev because sequential LLMs are “totally useless for computers”*, The New Stack.
- *TypeSafe AI Releases Jev*, MarkTechPost, 19 September 2026.
- *What Is Jev? TypeSafe AI’s System One Model Explained*, You.com.
- *Jev, Sorted: What Is Still Just a Claim*, Pere Pages, 16 September 2026.
- *Spring AI and TypeSafe Jev*, Spring blog, 21 September 2026.
- *How to classify, route, and score with Jev and AI SDK*, Vercel Knowledge Base.
- Oppor gateway: evaluations, Oppor documentation.
- *Awesome Jev* wiki, community reference.
- *TypeSafe AI’s Jev: What “System One Models” Actually Are*, TrueFoundry.

This paper describes a product that entered early access on 15 September 2026. Pricing, version policy, hosting and documented behavior may all change. Figures attributed to TypeSafe are the vendor’s own published claims and are labeled as such throughout; where a claim is self-graded or untested by an independent party, the paper says so. Nothing here is an endorsement of, or a recommendation against, any product.

Provided for general information; not legal advice. AI Risk Consulting is not a law firm.



TURNING AI RISK INTO ADVANTAGE

Governance that keeps pace with the systems it governs.

AI Risk Consulting works with banks, fintechs and other regulated organizations on AI and GenAI governance frameworks, model risk management, advisory and implementation, and independent research.

airisk-consulting.com

info@airisk-consulting.com