Jev is System 1.
Don't we still need System 2?

Autonomy Platform · Public Blog

PERSPECTIVE

© 2026 Azirella Ltd. All rights reserved worldwide.

Read more at azirella.com/blog

Technical September 2026

TL;DR

TypeSafe named Jev after Kahneman's System 1, and the name is more accurate than they may intend: every large language model is System 1, and Jev is simply the fastest. Nobody has said what plays System 2, the slow layer that checks a fast, confident answer. In a supply chain, where the world keeps moving, that checking layer is the product.

KEY POINTS

- Every LLM is System 1, with the biases Kahneman catalogued: it answers from what is in front of it, swaps a hard question for an easy one, and is as confident as the story is neat. Jev is the fastest one, and thinking longer does not cure it.
- In our architecture two things play System 2, and neither is a model: a range that widens when the world moves, and a gate that asks where a number came from rather than how big it is.
- "Can you name the answers first?" Supplier emails and override reasons pass. Adam DeJans's stockout-at-reorder-point example fails: it is arithmetic on an unknown value, which Jev's own explainer says to keep in code.
- The evidence is in TypeSafe's own caveats: the 0% hallucination figure is "not empirical", calibration is graded against two frontier models, and off familiar data the confidence broke while the accuracy held.

© 2026 Azirella Ltd. All rights reserved · azirella.com

Three pieces about Jev landed on my desk in the same week, and reading them side by side taught me more than any one of them did.

The first was TypeSafe's launch post. The second was Adam DeJans Jr.'s summary of why it matters, which has done a lot of the travelling on LinkedIn. The third was Avi Chawla's explainer in *Daily Dose of Data Science*, which is the calmest of the three and, I think, the most useful, because it quietly says out loud most of what the other two leave out.

That turned out to be the story. The caveats that matter were published by TypeSafe themselves, and the careful coverage kept them in. It was the enthusiastic relay that dropped them.

Start with the best sentence in Adam's post, because I agree with it completely:

> *"a model optimized on human preference has no particular incentive to say 'I am 61% confident' when 61% is genuinely the honest answer."*

That is exactly right, and the person behind the product has unusual standing to say it. Diogo Almeida is the fourth author of *Training language models to follow instructions with human feedback*, the 2022 InstructGPT paper. He helped show that preference optimisation works, and then concluded it optimises the wrong thing. He has since called the whole ChatGPT era a weird detour from real automation. Whatever else is true, that is not a critique from someone who missed the wave.

I should say where I sit. We build planning agents, and for two years we have held that a language model never makes the decision. That is a rule enforced at code review, not a preference. So Almeida and I are arguing adjacent things from opposite ends of the same problem. This is a disagreement between people who broadly agree.

They named it System One. What plays System 2?

Start with the name, because TypeSafe chose it on purpose. It is the first thing the launch post tells you:

> *"We were inspired by Daniel Kahneman, Thinking, Fast and Slow. The model class name draws on the distinction between fast, intuitive System 1 thinking and slow, deliberate System 2 reasoning."*

It is a good name, and I think it says more than they meant it to.

© 2026 Azirella Ltd. All rights reserved · azirella.com

In *Thinking, Fast and Slow*, System 1 is the part of your mind that answers instantly. It is fast, automatic and effortless, and it is usually right enough. It is also the source of nearly every bias in the book, because it produces an answer at once and with more confidence than it has earned. System 2 is the slow, effortful part that can stop, check the working and overrule the first answer, and Kahneman's point is that it is lazy and mostly does not bother. He never said fast thinking was good. He said it was unavoidable, and that the hard part is the checking.

Here is the thing about the name, though. Jev is not the first System 1 model. It is the fastest one.

Every large language model is System 1 in Kahneman's sense. It produces the most plausible answer to what it has just read, fluently and at once, and nothing inside it has the job of stopping to ask whether that answer is right. Jev strips out the writing and keeps the judgment, which is why TypeSafe can claim it is tens to hundreds of times faster. That is a genuine improvement, and I come back to what it gets right in a moment. But a faster System 1 is still System 1.

Why does that matter? Because Kahneman spends most of the book on what System 1 gets wrong, and his list reads like a description of how a fast model fails. Four of those biases matter here.

What you see is all there is. Kahneman's shorthand for it is WYSIATI. System 1 builds the most coherent story it can from whatever is in front of it, and never asks what is missing. A model does the same. It answers from the state you send it, and it cannot tell you that the one fact that would have changed its answer was never in the prompt. Chawla warns that irrelevant context hurts accuracy. The worse case is relevant context that is not there, because nothing flags an absence.

Substitution. Faced with a hard question, System 1 quietly answers an easier one and does not notice the swap. Asked how happy you are with your life, you report your mood this afternoon. Ask a model whether a SKU will stock out and the question it can actually answer is whether the text reads like the cases where stock ran out. Those sound alike and are not the same question, and the answer comes back just as fluent either way.

Judging by resemblance. System 1 judges how likely something is by how well it fits a familiar pattern, and ignores how common the thing actually is. That is the lesson of Kahneman's famous Linda problem. A classifier that sends a ticket to billing because it resembles billing tickets, whatever the real problem turns out to be, is doing exactly this, and it is one way a login bug ends up in billing.

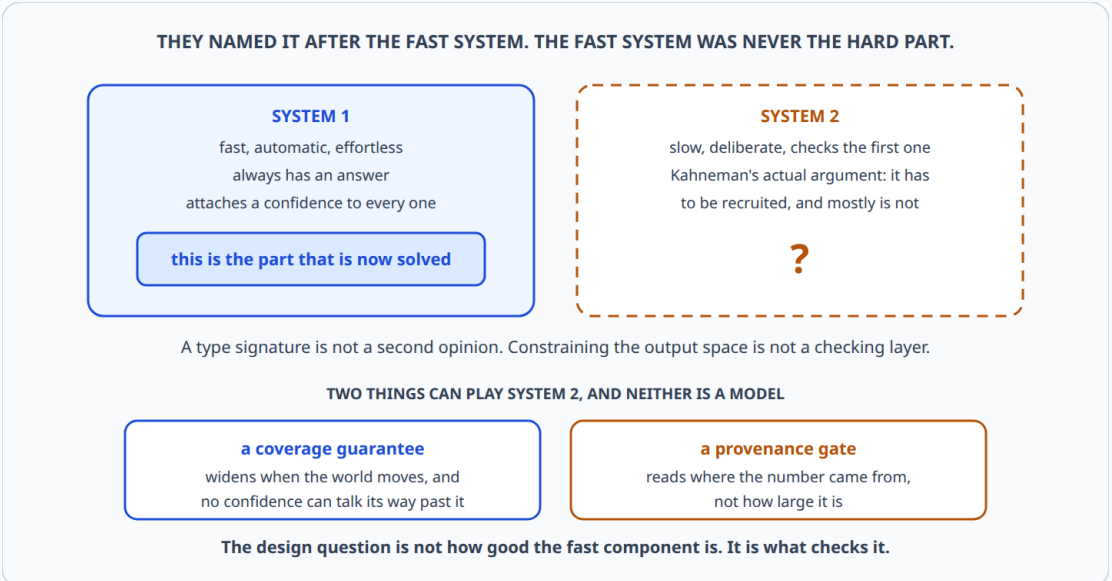
© 2026 Azirella Ltd. All rights reserved · azirella.com

Confidence that tracks the story, not the evidence. In Kahneman's words: "Confidence is a feeling, which reflects the coherence of the information and the cognitive ease of processing it." A model's confidence behaves the same way. It measures how neatly the answer fits, not how much evidence stands behind it.

None of these is a bug someone forgot to fix. They are the price of being fast. System 1 is quick precisely because it does not stop to ask what is missing, whether it answered the right question, or how common the thing really is. A faster System 1 pays the same price, only faster. Training Jev to be calibrated goes after the fourth bias, and on familiar data it works. It does nothing about the first three.

The so-called reasoning models do not escape this either, whatever the marketing says. A model that thinks for longer before it answers is still one model marking its own homework, with the same weights that wrote the first draft doing the checking. More time does not cure these biases, because the process doing the checking is the one that has them. That is System 1 given more time. It is not System 2, because what makes System 2 useful in Kahneman's account is that it is a different process from the one it checks.

So the name raises a question that the launch post never answers: what plays System 2? The nearest it comes is that your own code limits what the model is allowed to say. That is true, and it is not a checking layer. A type signature is not a second opinion. Chawla gets closer: "Its scores can inform an enforcement boundary, but they should not become one." That is the right instinct. The open question is what the boundary is made of.



© 2026 Azirella Ltd. All rights reserved · azirella.com

Here is the short version of my answer, which the rest of this piece sets out to earn. A confidence number that a model learned in training is calibrated on the world it was trained on, and a supply chain never stays in that world for long. So whatever checks the fast answer cannot be another learned number of the same kind, from Jev or from any other model.

In our architecture two things play System 2, and neither of them is a model. One is a coverage guarantee, from a method called conformal prediction, that widens the range around an answer when the world moves, and no amount of confidence can talk its way past it. The other is a gate that looks at where a number came from rather than how large it is. I come to both below.

And there is a third consequence, which is what makes the engineering worth doing. In our architecture the range is not something printed on a report. It is the instruction. Each planning level hands the level below it a target, the middle of the outcome its own plan implies, and a fence, the tenth and ninetieth percentiles of that same outcome, that it has to stay inside. That is the whole directive. A single number cannot do that job, because a single number has no fence, and a level handed one has been told what to do rather than what it is answerable for. Uncertainty stops being a caveat in small print and becomes the thing that is actually passed down. It is also the sharpest form of the shape problem I come to later: something that returns one number, however well calibrated, cannot hand anybody a directive.

What Jev gets right

None of that makes Jev a bad idea. Chawla opens with the problem it solves, and it is hard to put better: “Software makes thousands of small judgments: Is this ticket urgent? Which model should handle this request? Is this shell command dangerous?”

Most teams send each of those to a general-purpose language model, which writes out a sentence, one token at a time, before it gets round to the word “billing”. Then somebody writes code to parse the sentence, and more code to retry when the parse fails. For a yes-or-no question that is a lot of machinery.

Jev skips all of it. Text goes in, a typed answer with a probability comes out, and the questions are answered in parallel. TypeSafe claim 70 to 500 milliseconds and output that is not metered. I have not tested any of that, so take it as described, but the architectural point stands either way: something you consult on every SKU at every site every night lives in a completely different world from a chatbot a person is waiting on. Most of this industry has not caught up with that.

© 2026 Azirella Ltd. All rights reserved · azirella.com

We got to the same place from the other side. Our decision models are deliberately tiny and answer in milliseconds. Partly that is volume. Mostly it is that a small model cannot memorise the surface of a problem, so it has to learn the rule.

Small, fast, typed and parallel is right. My disagreement is entirely about what keeps an eye on it once it is running.

Adam's first example is the one to keep away from it

Adam opens with three judgments he says sit underneath every serious inventory or pricing decision:

> *"what is the likelihood this SKU stocks out at this reorder point, what is the confidence behind this demand classification, how sure are we that this supplier delay is transient rather than structural"*

Walk the first one through slowly, because it is where the argument comes apart.

Say a distributor has a SKU with a reorder point of 400 cases. The supplier takes about nine days, and the product sells around forty a day, though never evenly. Whether it stocks out depends on two things: how much you actually sell over the lead time, and how long the lead time actually turns out to be. The answer is a number you calculate from those two distributions. It moves when the reorder point moves, when the supplier gets slower, when a promotion lands.

Now look at Chawla's list of when not to use Jev. "Do arithmetic, counting, date comparison, and exact string manipulation in code." And a few lines later: "It cannot extract an unknown value directly. Find candidates first, then let Jev choose among them."

A stockout probability at a reorder point is arithmetic on an unknown value. The friendliest explainer of the model is telling you not to hand it that job.

His third example is the opposite. Whether a supplier delay is transient or structural usually arrives as words: an email from the supplier's account manager, a port advisory, a note a buyer forwards. The possible answers are known in advance, and a person could label a hundred of them in an afternoon. That is exactly the shape Chawla describes as a good fit, and I would happily use a model like Jev for it.

© 2026 Azirella Ltd. All rights reserved · azirella.com

So my problem with the post is not the enthusiasm. It is that it files reading and arithmetic under one heading, “probabilistic judgment”, when they need completely different machinery.

Where something like Jev fits

Chawla gives the best test for this I have seen, and it fits on a sticky note: “Can you name the answers first?”

Run a few supply chain questions through it.

Does this supplier email announce a delay? Yes, you can name the answers, and a person can label a hundred of them quickly. Good fit.

Is this planner’s reason for overriding the agent about a promotion, a quality hold, or a phone call from a customer? Good fit, and it is a job we genuinely have. Our work on whether overrides actually helped is starved of exactly that signal, because the reasons arrive as free text.

How many cases should we order? No. That is an unknown value and arithmetic, and Chawla has already told you to keep both in code.

Which SKU do we short when the line goes down? You can name the answers, but not in the space available. According to the published integration docs, a pick-one question takes at most 255 options and the whole situation has to fit in about 32,000 tokens of text. A mid-sized distributor carries thousands of SKUs, and the situation that decides which one to short is a network, not a paragraph. The decision, as posed, does not fit through the door.

Notice where all the good fits sit. Next to the decision, never in it. The launch post’s own comparison table puts Jev in the same place, under “classification, routing and scoring embedded in production code”. It does not say “making the decision”, and it is right not to.

I would be glad to see a model like this in a planning stack, on terms. A producer the customer chooses, speaking the same vocabulary as everything else in the network, and required to fill in the same four fields every decision in our system carries: what prompted it, what was decided, what outcome is expected, and the Achievability of that outcome, with the source of that Achievability declared. On those terms its numbers can be checked against what actually happened rather than against another model’s opinion, and if it beats what we have on a fair comparison it should win. I would say so.

© 2026 Azirella Ltd. All rights reserved · azirella.com

What I would refuse is the same model dropped into the decision itself, with no declared source and no obligation to explain, because it is small and fast and typed.

Which brings me back to one phrase of Adam's. He describes the opportunity as "the kind of 'smart if-statement' work that sits underneath inventory optimization, fraud detection, or automated procurement". An if-statement is something a person can read. A learned model whose output picks which branch runs is not an if-statement, it is a black box in the middle of the decision, and whether or not it technically writes text changes nothing. Anything that picks the action owes you a calibrated probability, a declared source, a reason you can inspect, and an outcome somebody measured. A schema fixes the type error. It does nothing about the accountability.

Check where the number came from, not how big it is

The second half of System 2 is less glamorous, and it is the part that matters day to day.

If your defence against badly calibrated probabilities is to review probabilities, you have signed up for work that grows every time anyone adds a new model, and you will lose. Eighteen months from now somebody wires in a new model. It produces a number between zero and one, and every check downstream accepts it, because a number in range looks like any other number in range. That is not a hypothetical. It is exactly how a confidence number in our own system that training never touched survived as long as it did, a story I tell in full below. It was always in range.

So we check the source instead. In our system every decision's probability has to say where it came from, and that label has teeth: a probability that did not come from the conformal layer cannot clear the bar for acting without telling anyone, however confident it looks. Not a higher bar. A ceiling.

Two details make that work rather than just sound strict.

If a producer does not say where its number came from, we assume the worst. A producer that will not tell you where its number came from is one nobody can check, and treating silence as the risky case is what protects you from the models nobody has written yet. That is the only kind of protection that scales.

The ceiling means "tell someone", not "stop". An unchecked number can still drive an action. It just cannot drive one that nobody hears about. That is why the rule is defensible rather than merely cautious. It takes away silence, not autonomy. Nothing gets slower. Things stop being invisible.

© 2026 Azirella Ltd. All rights reserved · azirella.com

The evidence, in four parts

That is the argument. What follows is why I believe it. There are four objections, and the first three are caveats TypeSafe published themselves: that a valid answer can still be wrong, that calibration has to be measured rather than claimed, and that their measure of it is graded against other models. The fourth is the one that matters most in a supply chain, and it is why the checking layer has to exist at all.

One: a valid answer can still be the wrong answer

Adam writes that Jev is

> *"a model that cannot hallucinate in the traditional sense, because its output space is constrained by schema rather than by the model's own restraint"*

Here is TypeSafe's own launch post, on the 0% hallucination figure in their charts:

> *"Our number is not empirical. Schema matching is guaranteed, thus we can confidently add 0% into the plots."*

That is unusually honest. They are telling you the zero comes from the definition, not from a test. The Register made the same point in its write-up, and Chawla put it in one line that belongs on a poster: "Type safety prevents malformed output, not bad judgment."

His illustration is the one worth remembering. A customer writes in about a login problem. It is a technical issue. Jev routes it to billing. Billing is on the list, so the answer is valid, the schema is satisfied, and nothing downstream objects. The ticket ends with a refund going to the wrong customer.

Nothing broke. Every field was the right type. The old kind of hallucination, an invented fact or broken JSON, is the kind you catch the same day. This kind you find at month-end.

I know this one from the inside, because we shipped it.

Several of our decision models used to show a confidence number beside each decision. It was always between zero and one, always well-formed, and it looked like exactly what you would want an agent to tell you. The training never touched it. Nothing in how those models learned ever rewarded that number for being right. It was a leftover from how the network happened to be initialised, dressed up as a probability.

© 2026 Azirella Ltd. All rights reserved · azirella.com

When we found it, we tried the obvious fix: use how strongly the network preferred the action it chose. That read 0.99 or higher on between 82 and 86 percent of situations. Of course it did. How firmly a model prefers one lever over another says nothing about whether pulling it works.

Every one of those numbers passed every type check we had. What we took away was a rule rather than a patch: nothing inside the decision model gets to supply the decision's probability, because nothing in there is measuring the right thing. The probability has to come from somewhere else, and it has to be checked against what actually happened. That is what the range and the source check described above are for.

Two: calibration is something you measure, not something you claim

This is the friendly ask.

Calibration is not a feeling. There is a standard test and it is simple. Group the predictions by the confidence stated, then check how often each group came true. Things said at 60% should happen about 60% of the time. Draw that and you have a reliability diagram. The launch post does not include one. For a product whose whole pitch is calibration, that is the one chart I most wanted to see.

Chawla's worked example shows why it matters. A ticket comes back with billing at 0.52, technical at 0.46 and sales at 0.02. His verdict: "Billing won by six points. Automatic routing would be reckless."

Quite right. But look at what you are relying on when you read that. You are trusting that 0.52 really means something close to a coin toss. If the model's numbers run hot, that close race was really a landslide, and if they run cold it was the other way round. Either way, the policy you built on top of it is standing on sand. Which is why the sentence that follows in his piece is the one that matters:

> *"That relationship must hold on your traffic. Measure it separately for each decision type and recheck it after changes to the model, questions, or input distribution."*

That is exactly the right advice. It is also a job, and every customer has to build the instrument for it themselves.

Someone outside the company has already built one, four days after launch, for about six cents, and published all 4,621 raw responses. They ran three public benchmarks and one task the model could not have seen: 900 rule-generated support tickets, scored

© 2026 Azirella Ltd. All rights reserved · azirella.com

against a noise floor worked out by simulating a perfectly calibrated model two hundred times.

On the public benchmarks Jev came in at one to 1.7 times that floor. Expected calibration error of 0.024, 0.032 and 0.029. That is a good result and I want to say so plainly: on the kind of data it was built for, the calibration claim held up. Nobody can rule out that the public benchmarks leaked into training, so as the author says, those rows are context rather than proof. The interesting result is the task nobody could have trained on, and I will come back to it.

Three: graded against two other models

This is the one that stopped me, and it matters well beyond this product.

Here is how TypeSafe's workflow evaluations decide what the right answer was:

> *"use the predictions of the largest, smartest, and most expensive external models as reference probabilities"*

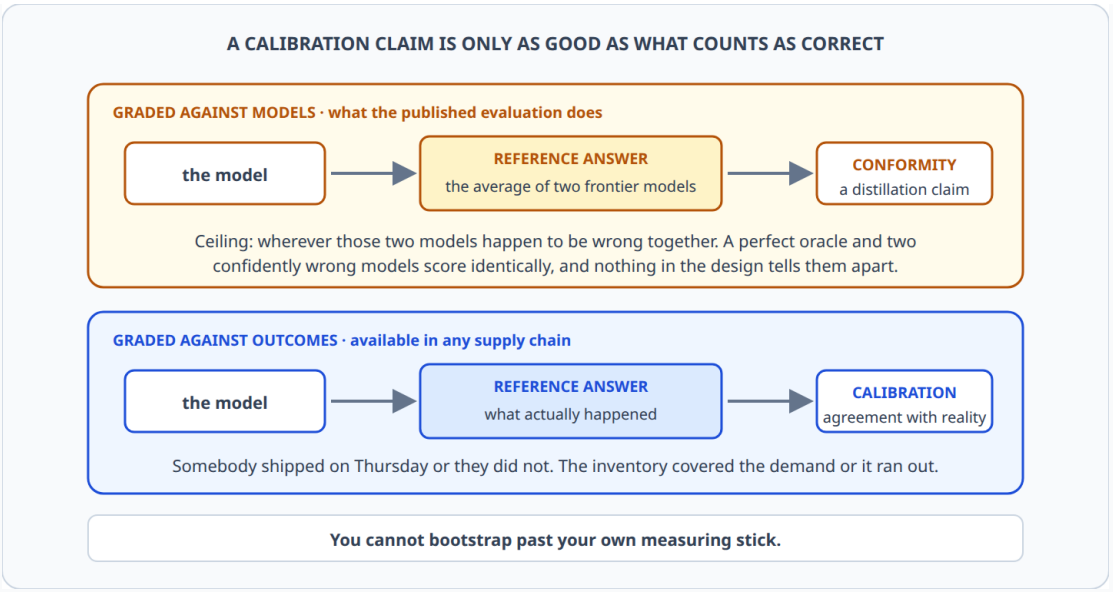
In the worked examples that means "the average of GPT-6 Astra and Fable 5.1 as the reference answer", which, to their credit, they note "biases answers towards OpenAI and Anthropic's models."

Picture a trainee demand planner. You want to know whether her forecasts are well calibrated, so every month you compare her numbers with the average of your two most senior planners. A year later she matches them almost perfectly. What have you learned? That she thinks like them. You have learned nothing about whether any of the three of them is right, and in the months when both seniors were wrong in the same direction, she scored full marks for being wrong too.

That is what this evaluation measures. It is a legitimate design, and a lot of the field scales its evaluation exactly this way. What it can show is distillation: a small model reproducing the judgment of much larger ones, far faster and far cheaper. That is a real and valuable result, and if it holds it justifies the funding round on its own.

What it cannot show is calibration. Calibration is agreement with what happened. This is agreement with what two models thought. A model that matched a perfect oracle would score the same as one that matched two confidently wrong models, and nothing in the test tells them apart. There is also a quiet irony here: the reference is two models of exactly the kind the launch argues have no reason to be honest about probability.

© 2026 Azirella Ltd. All rights reserved · azirella.com



In our line of work there is no excuse for that shortcut, because the right answer turns up by itself. Somebody shipped on Thursday or they did not. The stock covered the demand or it ran out. When the answer is sitting in your own transaction history, grading against a model's opinion is a choice.

Four: calibrated on the world that was

In February 2020 a lot of demand models in this industry were well calibrated. By the end of March almost none of them were. Nobody's code had changed. The world had.

That is the real problem, and it is why none of the objections above is the deepest one. A model trained to be calibrated is calibrated on the data it was trained on. That is the only thing its training can see. So it is, quite precisely, calibrated on the world that was.

In most domains that costs very little. In a supply chain it is close to the whole problem, because the world here moves by design, not by accident. A plant opens. A supplier gets dual-sourced. Three distribution centres become one. A tariff changes the landed cost of a whole category. A competitor launches something and a demand pattern changes for good. There is never a long quiet stretch to be calibrated against. Chawla, to his credit, tells you to recheck after the input distribution changes. In our world that recheck never ends.

This is where the six-cent test earns its keep. On the 900 tickets the model could not have seen, expected calibration error came in at 0.107 against a floor of 0.024. That is four and a half times the floor, against one to 1.7 times on the public sets. By question type it is worse than the average suggests: 0.082 on the pick-one questions, 0.079 on the yes-or-no ones, and **0.325 on the ordered-score question**.

© 2026 Azirella Ltd. All rights reserved · azirella.com

That score question is worth a closer look, because it is the first bias from the top of this piece, measured. It asked for a ticket's priority, and the right answer depended on an organisation's policy that was nowhere in the text. The model could not have known it. It answered anyway, got under half of them right, and reported an average confidence of 0.74. What you see is all there is.

Then the part that should stop you. **The accuracy held up.** Around 75% overall, and 89 to 92% on the questions that were actually answerable. The model went on being roughly right and stopped knowing how right it was.

That is this whole section in one sentence. When the world moves, the first thing to go is not the answer. It is the confidence attached to the answer, and the confidence is the exact thing you were planning to automate on. Set your threshold using the numbers from the familiar data and you have set it for the wrong world.

The errors even change direction by question type. The pick-one and score questions came out overconfident and the yes-or-no ones underconfident, which is why the author's conclusion is sharper than any single verdict on the model: check calibration per question, not per model.

Those numbers carry caveats the author is careful to state, and they are the reason to trust the rest. One synthetic family of tasks. 900 items. The model version not visible through the API. And an explicit instruction to re-run the test before quoting it later. I am quoting it four days on, so read it as a snapshot of a fast-moving thing, not a verdict.

This is why our uncertainty layer is built on conformal prediction rather than on a trained confidence number. Conformal gives a guarantee that still holds when the model underneath is wrong, because it comes from how the method is built rather than from the model being any good. What that means in practice is the whole point: **a worse model earns a wider range, not a wrong one.**

Think of it as the difference between a fuel gauge that sticks at half and one that starts to flicker. The flickering one is annoying. The stuck one runs you out of fuel on the motorway. A model whose world has moved should flicker: it should report a range so wide it is obviously useless, and an obviously useless range is an alarm. It turns silent failure into loud failure.

There is a lovely result from Keyon Vafa and colleagues at ICML last year. They trained a model on orbital trajectories and it predicted orbits beautifully. But when they asked it to apply what it knew to new physics problems, it did not use Newton's laws. It had learned Kepler: a very good description of the data it saw, and no theory of why. A confidence number tuned on one fixed world is the same bargain in a smaller package.

© 2026 Azirella Ltd. All rights reserved · azirella.com

None of this argues against training for calibration. Do both. Conformal wraps whatever you give it, and a better-calibrated model earns a narrower range for the same guarantee, which is worth real money. The argument is only against letting the trained number carry the whole weight, because when it fails, you cannot see it fail.

One number, two kinds of not knowing

Adam's post says the confidence signal comes "Returned with every answer, and calibrated", against a status quo he fairly describes as "overconfident and inconsistent when prompted for one". Half of that is right. A confidence you only get when you remember to ask for it is not really a confidence signal.

But hold it up against his own test of what matters:

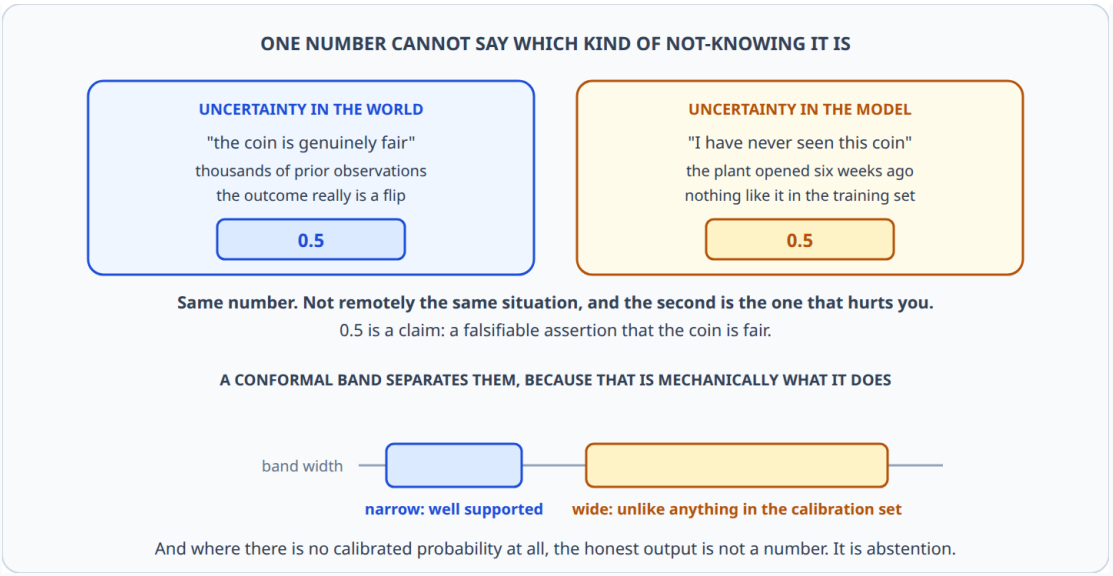
> *"The one that matters for automation is being fought over who is most honest about what they do not know."*

One number cannot carry that, and the reason is practical, not philosophical.

Here is an illustration. A distributor opens a new distribution centre in August. In September a model is asked whether a particular delivery will arrive on time, and it says 0.5.

That 0.5 could mean two completely different things. It could mean the lane really is a coin toss: there are two years of records and half the loads arrive late. Or it could mean the model has six weeks of history on a site it has never seen before and simply has no idea. Same number. The first is information. The second is a guess that looks exactly like information, and it is the one that hurts you, because 0.5 is a claim. It says the coin is fair, and that is only honest once you have actually checked the coin.

© 2026 Azirella Ltd. All rights reserved · azirella.com



This is where conformal earns its place a second time. Its range widens when the case in front of it looks unlike anything it was calibrated on, because that is simply how the method works. A confidence read off the shape of the model's own output cannot do that. It tells you how sure this answer sounds, not how much evidence there was behind it.

And underneath that, in our system, having no calibrated probability is treated as its own situation, deliberately not as zero and not as a half. Zero is a probability, and if you score "we do not know" as a confident zero, any other term in the formula can outweigh it. I know that because our first version did exactly that. Now a missing probability takes a separate path: the system declines to vouch for its own answer, sends the decision to a person, and records it as "did not know" rather than "low confidence", because those are different events and an auditor should be able to tell them apart.

That is a designed outcome, not a failure. A system that can never say "I don't know this one" is not more autonomous. It is less honest, and it has one fewer way of being right.

The number is not the point

I said at the start that the caveats were TypeSafe's own and that the careful coverage kept them in. Chawla's piece keeps every one: the zero is not empirical, calibration has to be measured on your own traffic, scores should inform a boundary and not become one, and "closed weights, early access, and limited independent calibration data make blind trust premature." It was the enthusiastic relay that dropped them. That is worth saying plainly, because the problem in this industry is rarely vendors lying. It is careful claims getting passed along until the care falls off, and then getting bought.

© 2026 Azirella Ltd. All rights reserved · azirella.com

Almeida is right that aiming for honest probabilities instead of pleasing answers is the shift that matters, and he has better reason than most to know what the alternative cost. I would publish the reliability diagram. I would grade against what actually happened rather than against two models agreeing with each other. And in a business that will not hold still, I would not ask a trained number to carry the whole guarantee on its own.

Chawla ends his piece like this: “An if statement handles explicit values. Jev estimates the meaning needed to choose among them. The application decides whether that estimate is safe to act on.”

That last sentence is the one I would underline. In a supply chain, deciding whether an estimate is safe to act on is not the plumbing around the product. It is the product.

They named it after the fast system. The fast system was never the hard part. Kahneman’s whole argument is that it is the part you get for free and cannot switch off, and that everything worth having depends on what checks it.

The number is not the point. Where it came from is.

See Autonomy in action

Walk through how Autonomy models, executes, monitors, and governs supply chain decisions with autonomous AI agents.

See It Live

© 2026 Azirella Ltd. All rights reserved · azirella.com