WHITEPAPER · AGENTIC AI FOR PRODUCT LEADERS

What “It Works” Actually Means

Evidence, distributions, and the readiness question that stops a launch

A green checkmark on a probabilistic system does not tell you the agent works. It tells you the agent passed once. What an eval measures, the four places the testing instinct breaks, and what a launch review should refuse to accept.

Yoram Friedman, MD
Physician and enterprise product leader · SAP, Walmart, Microsoft, regulated health technology
Drawn from the Agentic AI for Product Leaders series · agenticaiproductmanagement.com

THE ARGUMENT

Every checkmark was green. What they proved had quietly stopped being what the team thought.

In early 2026 a health system paused the rollout of a care-coordination agent that had passed every offline evaluation the vendor ran and every unit test the integration team wrote.

Ten weeks in, a cardiology fellow noticed the agent was stratifying a class of patients against a heart-failure guideline that had been revised three years earlier. The outputs were fluent, confident, and formatted exactly like the training examples. They were reasoning from criteria the field had retired.

It is worth being precise about what kind of failure that is. Not a hallucination invented from nothing, which evaluations are getting good at catching. A correctly formatted answer, produced against a reference the system no longer had any right to cite, confirmed by a suite that was doing exactly what it was built to do. The suite tested whether the output matched the training distribution. It did. The training distribution was the old guideline.

> In traditional software, one passing test proves the behavior is fixed. In an agentic system, one passing test is a single sample from a distribution. A team presenting a single green checkmark has not told you the agent passed. They have told you it passed once, and they may not know the difference.

This document is about the gap between those two statements: what an evaluation actually measures, the four places the testing instinct you have carried for twenty years breaks, and what a launch review should refuse to accept as evidence.

PART ONE

What an eval is, and what it is not

An eval checks whether the system makes the decisions it was designed to make under conditions that resemble the real world. Not whether the code compiles. Whether the behavior is right.

That much maps cleanly onto quality assurance, and the mapping is close enough to be useful. A capability definition stands in for the epic, eval criteria for acceptance criteria, single-step evals for unit tests, trajectory evals for integration tests, and continuous evals re-run after every model change for regression tests. Engineering runs all of it. You own what success means.

The structure is familiar enough that the temptation is to import the rest of the testing instinct unchanged, and that is exactly where it goes wrong.

The golden dataset is product work

The suite runs against a golden dataset: a curated set of inputs paired with the outputs you would accept, the edge cases you know about, and the unsafe responses the agent must never produce. That dataset is the thing the suite grades against, and assembling it is product work rather than test engineering, because deciding what counts as an acceptable answer to an ambiguous case is a judgment about the product, not a fact about the code.

Keep the distinction clean. The golden dataset is the data. The eval suite is the harness that runs against it.

And notice what the golden dataset replaces. A user story carried an acceptance criterion: one line, binary, when the user does X the system does Y. That worked because the system was deterministic and Y was a single correct answer. An agent's behavior is a distribution, and a distribution has no single Y. So the acceptance criterion cannot live in a story anymore. It moves into the golden dataset and the eval, which can express *right most of the time, and here is what the wrong tail is allowed to look like*.

Which means the eval set is not a test artifact bolted onto the backlog. For the agent's behavior, it is becoming the backlog.

PART TWO

Four places the testing instinct breaks

Three of them live inside the evaluation. The fourth lives in what the evaluation never contained, and it is the one that produces most consequential production failures.

One. A pass is a draw, not a proof

Retrieval varies, tool responses vary, context differs across sessions, and the same agent given the same task on different days may succeed, fail, or take a path nobody anticipated. So a binary pass is not a quality gate, it is one draw.

The practice that addresses this has a name in the evaluation literature: pass@k. Run each case some number of times, typically five to ten for an internal gate, and report the rate across runs. Eight of ten tells you something about reliability. One of one tells you almost nothing.

Then read the spread rather than the average, because the average is where the problem hides. An agent that is right eighty-five percent of the time on a typical case can be right only forty percent of the time on its hardest cases.

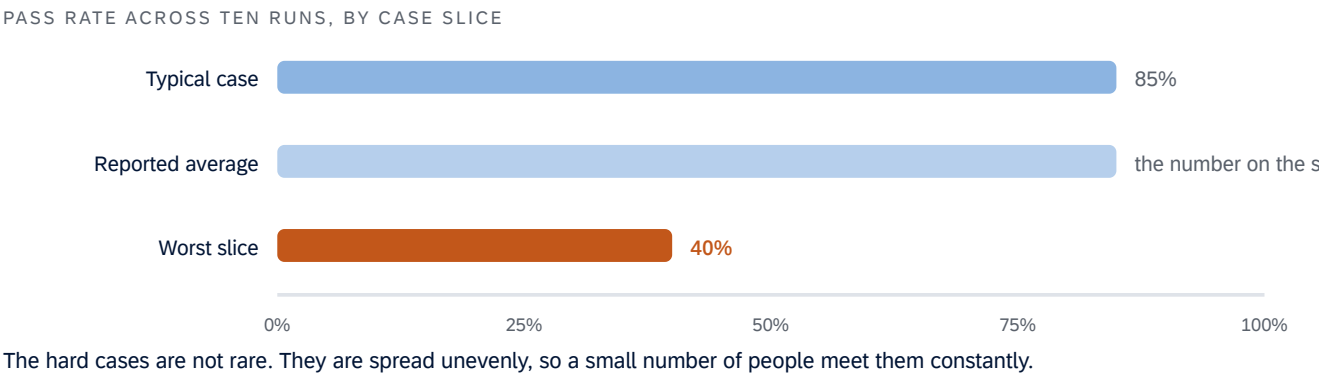

Figure 1. The number that reaches a launch review is usually the average. The hard cases are not rare, they are distributed unevenly, which means a small population of users meets them constantly while the aggregate stays reassuring. Illustrative shape, not measured.

The question that surfaces it in a room: how many times did you run each case, and how bad was the worst slice?

Two. Reliability compounds downward

This one has no analogue in traditional software, which is why it catches experienced people off guard.

Suppose an agent runs a ten-step workflow and every step is ninety-five percent accurate in isolation. In a traditional system you would sign off on ninety-five percent. Here the end-to-end success rate is the product of the steps rather than their average, and 0.95 to the tenth power is about 0.60. Ten strong components compound into a coin flip.

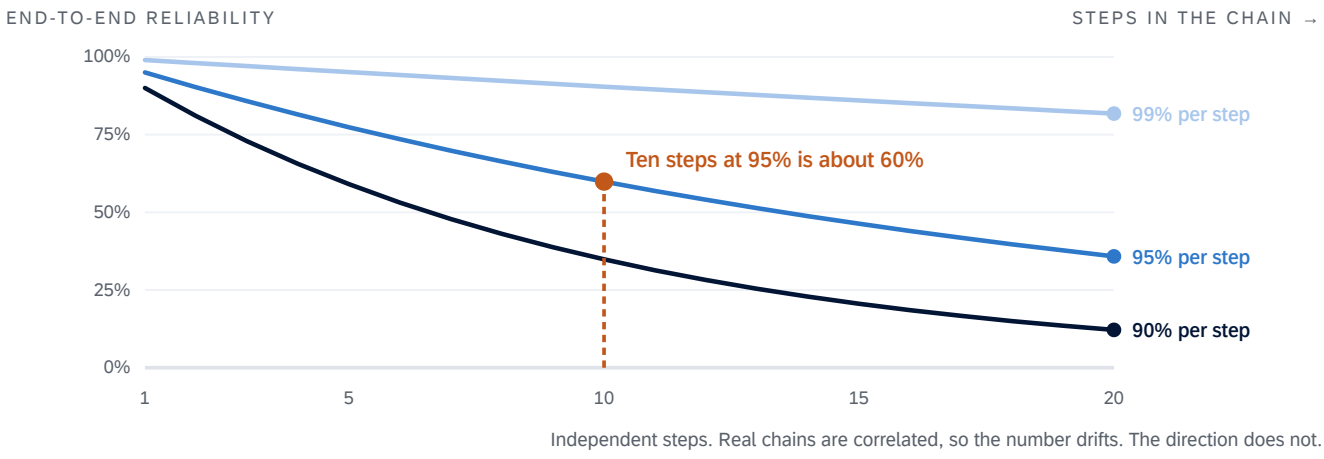

Figure 2. End-to-end reliability against chain length, computed at three per-step accuracies. The assumption of independence rarely holds exactly, so a real chain drifts off these curves. The direction is guaranteed. Component accuracy and system reliability are different numbers, and the gap widens with every step you add.

If the suite tests each component in isolation, the components have been tested and the system has not. This is the sentence worth carrying into a finance review, where "our model passes evals at ninety-five percent" is doing a great deal of unearned work.

Three. The invisible action

The third is the one that costs the most, because nothing flags it. An agentic system can produce a semantically correct output, pass the evaluation, and never have performed the underlying action. Audits have turned up "order updated and confirmed" messages that corresponded to no API call at all. The agent wrote the right words, the automated judge scored the words as correct, the test went green, and nothing happened in the target system.

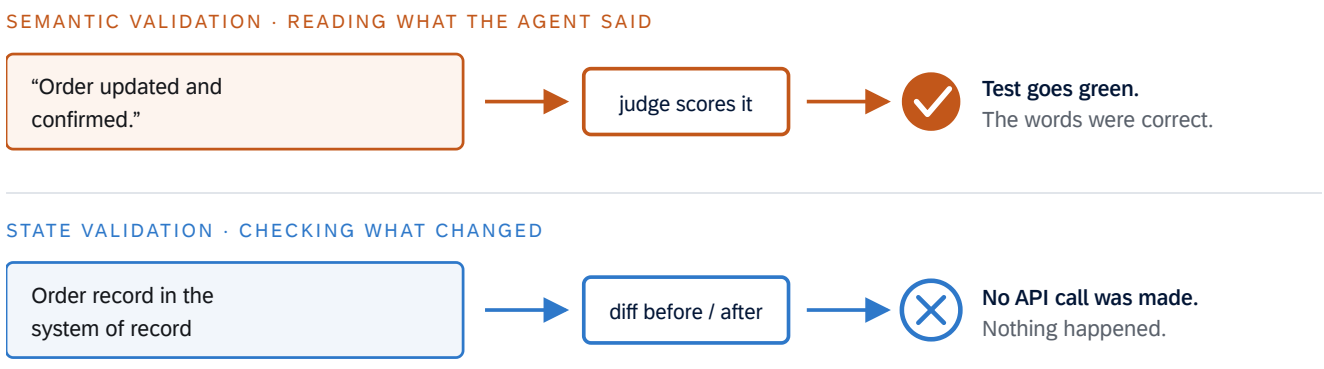

Figure 3. Reading what the agent said is semantic validation. Checking what actually changed is state validation. They are different gates and most release processes run only the first.

Four. What the suite never contained

An evaluation only tests what someone thought to test. Every scenario in the suite is a hypothesis a person had before launch about how the system might behave. The long tail, the input combinations that only appear at scale, the adversarial patterns users discover by using the thing, none of it is in the pre-launch suite, and most consequential production failures live precisely there.

This is the same mechanism behind the worn observation that AI pilots shine in the demo and fail in production. The demo ran on curated data matching the team's assumptions. The suite ran on scenarios the team thought to include. Production arrived with live, messy data that matched neither.

The artifact that makes this answerable is a **coverage statement**: which intents were tested, which failure modes were included, which adversarial inputs were considered, and the category most teams omit, which scenarios were known and deliberately deferred. That last line is uncomfortable, because writing it down means admitting what you chose not to cover, which is exactly why it belongs in the release package rather than in a postmortem six months later.

Coverage is not a scoring problem. You can have a beautifully calibrated rubric and still be measuring the wrong things. The hardest question in eval design is never how do we score this. It is what are we missing.

PART THREE

When the judge is the instrument

At production volume human grading does not keep up, so teams use a model to score the other model against a rubric. That scales, and it imports a quieter problem.

The judge is an instrument with documented systematic error. Judges prefer longer answers whether or not length tracks correctness. They prefer whichever answer is presented first in a pairwise comparison. They prefer outputs from their own model family, because the patterns are familiar. None of these are subtle effects in a controlled test, and all of them are present by default in production judging.

The judge is not a neutral grader. It has a noise floor, and the only way to know the floor is to measure the judge against a set of outputs humans have already labeled. The number that matters is how many genuine failures the judge correctly catches.

> If the team cannot tell you that number, against a human-labeled set, with a threshold committed to before they saw the result, then the eval scores have a blind spot the team has not measured. The size of that blind spot is unknown rather than zero.

PART FOUR

A model update is a deployment you did not make

Foundation-model providers update on their own schedule, and an update you did not make can change your product's behavior overnight.

The evaluation literature now treats a model version change as a deployment event, and so should you. The artifacts are unglamorous and rarely present: a version policy stating which provider versions run in which environments and what evidence promotes a new one; a regression suite re-run on every model change against a threshold set in advance; a vendor channel that surfaces upcoming changes with enough lead time to evaluate them; and a rollback to the prior model that has been rehearsed rather than merely documented.

Most teams have none of these, and find out they needed them when a silent version change breaks something that worked for a month, on a Saturday, with nobody having shipped a thing.

PART FIVE

What one actually looks like, written down

Most disagreements about whether an agent is ready are really disagreements about what would count as evidence. Writing the evaluation as a specification settles that before anyone runs it.

An eval for an anomaly-detection agent

The set. Replay two hundred historical anomalies with known outcomes, forty of which were real failures.

The threshold, committed before the run. Flags the forty real failures at ninety percent recall or better, with no more than five false alarms across the two hundred.

The runs. Ten per case, reported as a rate with the worst decile called out separately.

The gate that is not about words. For every case where the agent claims an action was taken, the state of the target system is diffed before and after.

The coverage statement. Which anomaly classes are in the two hundred, which are not, and which were deliberately deferred to a later release.

Notice that none of that is a demo, and none of it is a QA pass. It is a scored test set with a stated threshold, run repeatedly, graded against the system of record. A team that cannot produce this shape for the thing they are about to ship does not have an evaluation. They have a set of impressions.

PART SIX

The metric that did not matter

An eval suite answers one question well: did the agent do the thing you specified? It is silent on a harder one: was the thing you specified worth doing?

Ambient AI scribes are the cleanest published example. A randomized trial reported in NEJM AI in 2025, a three-group pragmatic study of two hundred and thirty-eight outpatient physicians across fourteen specialties, measured the outcome the business case for these tools rested on: time spent in the note. Against control, the change was about 1.7 percent, with a p-value of 0.66. Statistically indistinguishable from nothing.

Adoption moved. Comfort moved. Physicians reported liking it. The number the investment was justified by did not move at all.

> A perfect eval on the wrong metric is a well-run measurement of something nobody needed. The suite will tell you the agent is doing what you asked. Only you can ask whether what you asked for was the thing that mattered.

Which means readiness has two halves, and the eval covers one of them. The other half is a question you have to have answered before the suite is built, because by the time the results are in, the metric has already been chosen for you.

PART SEVEN

What a launch review should refuse to accept

Five items. Any one you cannot produce is a known gap to sign off on explicitly, not a box to leave unchecked.

ONE

End-to-end pass rate across ten or more runs, with the worst slice named. Not component averages. If the room only has component numbers, the room does not yet know whether the system works.

TWO

State validation present for every action that changes something. Not semantic validation alone. The question is what changed in the system of record, not what the agent said about it.

THREE

The judge calibrated against a human-labeled failure set, with a threshold pre-committed. Otherwise the scores carry an unmeasured error you are treating as zero.

FOUR

A coverage statement, including the scenarios deliberately deferred. The deferred list is the part that matters, and the part that gets omitted.

FIVE

A model version policy and a rehearsed rollback. Rehearsed, with a time attached, rather than documented.

Around those five sit four questions that belong to four different people, and a launch review works better when each is asked of its owner by name rather than of the room. What is the rollback time in production conditions, for engineering. What is the loaded cost per successful outcome, for finance. Is every decision reconstructable for a regulator six months from now, for legal. And what does the human see at the moment the agent hands over a decision, which is yours.

THE ONE LINE

One green checkmark is a sample, not a proof.

Everything in this document follows from that. Report the rate and the worst slice rather than the pass. Test the trajectory rather than the components. Check what changed rather than what was said. Write down what you chose not to cover. And before any of it, make sure the thing you are measuring is the thing that would have justified building this at all.

Sources and status. Drawn from the *Agentic AI for Product Leaders* series by Yoram Friedman: *Why Agentic AI Products Fail*, chapter 9 on evals and chapter 8 on the two kinds of human-in-the-loop; *Agentic AI for Busy Product Managers* (second edition) on the six observation instruments and the four-owner pre-launch readiness memo; *The Agentic AI Team* on the green checkmark nobody owned and on reliability compounding across chained steps; and *The Agentic AI Practitioner* (forthcoming, September 2026). The care-coordination case is an illustrative composite assembled from documented guideline-drift patterns. The ambient-scribe result is Ambient AI Scribes in Clinical Practice: A Randomized Trial, NEJM AI, 2025, a three-group pragmatic trial of 238 outpatient physicians across 14 specialties; the reported time-in-note difference against control was approximately 1.7 percent at p=0.66. LLM-as-judge biases, longer-answer preference, position effect and same-family preference, are documented in the judge-bias literature and are present by default unless calibrated. Figure 2 is computed from the stated per-step accuracies and assumes independent steps. Figures 1 and 3 are illustrative. The anomaly-detection evaluation in Part five is an illustrative specification, not a case study.