CONVIVA

A MARKET ANALYSIS

Why Agent Operations Needs an Experience Layer

And how we define what it is

Keith Zubchevich

President & Chief Executive Officer, Conviva

conviva.ai

SUMMARY

Enterprises deployed agents faster than they built any way to know whether they work.

Enterprises have deployed customer-facing AI agents faster than they have built any way to know whether those agents work. The measurement practices filling that gap were not designed for this problem. They were borrowed from two neighboring fields, software observability and machine-learning evaluation, and both are being asked to answer a question neither was built to ask.

The conversation can score well against every rubric it was graded on. Each response was accurate, on-brand, defensible. The customer left anyway, and nothing in the stack registered that they had.

The result is a market operating on instruments that report health while customers leave. This is not a vendor failure. It is a category gap, and category gaps need to get filled.

CONTENTS

This paper argues seven things.

1. **The borrowed methodologies fail structurally, not by accident.** Observability and evaluation are not immature at this job. They were designed to answer different questions, which means better versions of them will not close the gap. They are also scoped to a single span, a single turn, a single agent, at exactly the moment production systems stopped being any of those things. (Sections 1 and 2)

2. **Experience measurement has objective requirements, and almost nothing in market satisfies them.** There are six properties a metric must have before it is worth running an operation on. Naming them is more useful to this market right now than another dashboard, because it gives buyers a standard to measure vendors against. (Section 6)

3. **The alternative to LLM-as-a-judge is not a better judge.** It is the user as the judge. The party being served passes judgment constantly, without meaning to, through what they do inside the conversation and what they do around it. Reading both together is the only signal in this market that cannot be flattered, cannot drift, and does not depend on someone having anticipated the failure. (Section 3)

4. **The largest driver of token consumption in a production agent is bad experience.** Not the model, not the prompt, not the architecture. Wasted turns are billable, they pile up faster than the turn count, and they are invisible in every dashboard a team currently watches, because a wasted turn looks exactly like a successful one. (Section 8)

5. **Experience measurement is an instrumentation problem, not an inference problem.** This is the most consequential point in any build-or-buy decision. The difference is not that one side computes and the other infers. Everyone has deterministic compute. The difference is what it runs on. A rubric has to be written in advance, and nobody can write one for what real customers actually do, so in production their signal falls back to a model grading a sample. A conversation needs no rubric: who repeated themselves, what got dropped, where it stalled. Counting that once is trivial. Counting it statefully, across every dimension, on all traffic, while the conversation is still open is where architectures fail. And that failure arrives dressed as methodology. Sample this percentage. Score that slice. Review these traces. Those are not chosen methods. They are the shape of an immature stack, and they should be read that way. (Sections 3 and 4)

6. **Experience, not cost, should be the first vector in model routing.** Where multi-model architectures are the norm, experience should determine which model handles which class of request, with cost as the tiebreaker at parity. That reordering is worth more money than most organizations currently realize, and it is unavailable to anyone who cannot measure experience in production. (Section 9)

7. **The transition has a precedent close enough to be predictive.** The video industry made exactly this shift, in the same sequence, for the same reasons, and the outcome was not ambiguous. (Section 11)

SECTION ONE

The current state of agents: a category running on the wrong instruments

Every enterprise deploying agents chose a measurement stack that is not ideal. It is two disciplines that were already in the building, both mature, both credible, and both aimed somewhere else.

Software observability was built to answer whether a system is running. Availability, throughput, error rates, latency distributions, cost. It is excellent at this and remains necessary. But observability's entire design assumption is that a functioning system implies a served customer. For most of software history that was close enough to true.

Machine-learning evaluation was built to answer whether a model output is acceptable against a specification. It is a pre-production discipline: a regression gate, a release control, a way to ship without breaking what worked yesterday. Also necessary. Also aimed elsewhere.

Neither discipline is deficient. They are correctly built for their own purposes. The problem is that agents broke the assumption both of them rest on: that a technically correct system produces a satisfied customer. An agent can be available, fast, inexpensive, and defensibly correct at every individual step, and still deliver an interaction that ends a customer relationship.

> The system worked. The experience failed. Those are two different measurements, and the market only measures one of them.

SECTION TWO

Why the gap is structural, not a product-maturity problem

The most common objection to this analysis is that the incumbent categories are simply early and will extend into experience measurement as they mature. The evidence suggests otherwise, for reasons that sit below the product line.

System performance is lagging by construction. Infrastructure telemetry describes what already happened. It is a record, and a useful one. But it is not indicative of what a customer is about to do, because the customer's experience is not a function of the infrastructure's condition. This is the same reason the video industry eventually stopped managing viewer engagement from delivery metrics: the signals that told you a viewer was about to leave came from watching the viewer, not the system. No amount of refinement turns an after-the-fact system record into an early warning about a customer. They are different subjects of measurement.

Evaluation is bounded by what someone thought to test. An eval is a test written against an anticipated case on a curated set. That is precisely why it works as a release gate, and precisely why it cannot serve as a production truth layer; it is structurally blind to the asks, intents, context and failure nobody imagined. There is also an arithmetic problem underneath it that the market has been slow to absorb: reliability compounds down the whole chain of steps. A step that is 95% reliable, run twenty times, is a task that succeeds about a third of the time. Demos are three steps. Production is twenty-five. Step-level scoring will report health the entire way down, because every step really was fine. Reliability lives in the whole run. Evaluation is scoped to the step.

95%

RELIABILITY OF A SINGLE STEP

1 in 3

TASKS THAT SUCCEED WHEN THAT STEP RUNS 20 TIMES

3 vs 25

STEPS IN A DEMO VERSUS STEPS IN PRODUCTION

And the unit of failure has already moved twice, while the unit of measurement has moved once. Evaluation grades a span: the record of one step the system took. The market's more sophisticated answer grades a turn: one exchange between the person and the agent. The industry has spent the last year falsely excited about that move, from single-turn to multi-turn. The move was real. It was also in the wrong direction, because it made measurement longer when what changed is that it needed to get wider. Production is no longer one agent taking many turns. A single customer request now routes through an orchestrator, a retrieval agent, one or more domain specialists, a tool-calling agent, and something that assembles the answer. Each of those can be individually correct while the request as a whole fails.

No single agent failed. The combination did, and nothing a per-span or per-turn score computes can represent that. It lives in a handoff. In context that one agent held and the next was never given. In a specialist returning a confident answer the orchestrator had no basis to check. In two agents that were each right about their own piece and jointly wrong about the request. Every component passes its eval. The interaction fails. Scoring the components harder will never find it, because the components were fine, which is the same structural error as step-level scoring, one level up, and it compounds the same way.

The order of operations has to invert. Measure the experience end to end first, at the level of the interaction the customer actually had, and only then resolve backwards to the agent and the turn where it went wrong.

> Diagnosis runs from the whole to the part. Almost everything in market runs from the part and hopes the whole adds up.

And the industry's answer for production, putting a model in front of the output to grade it, inherits all of this while adding several problems of its own. That approach has become the default assumption about how agent experience gets measured, which makes it worth taking apart carefully rather than dismissing. It is also the decision that determines whether everything built on top of it can be trusted, so it is where this paper goes next.

SECTION THREE

Why this cannot be built on judged, sampled signal

The intuitive move, when the thing you are measuring is made of language, is to measure it with a language model. Put a judge in front of the output, give it a rubric, collect scores. It demos well, it ships in a week, and it is the assumption underneath most of what the market currently calls agent quality measurement.

One clarification before the argument, because it is routinely misheard. The claim is not that we compute and they infer. That would be false. Every serious tool in this market has deterministic compute in it: counts, latencies, token totals, error rates, pass rates, all exact and all reproducible.

The distinction is what each system computes deterministically, and where the experience signal comes from. Their deterministic layer computes system telemetry, and aggregate pass rates on results that a model judged. It works precisely and only where a well-defined rubric already exists, and a well-defined rubric exists in the lab, against a curated set, on the cases someone anticipated. Carry that architecture into production, where no rubric exists because the open world does not supply one, and the experience signal falls back to what a model decided, on whatever fraction of traffic the budget allowed. This is not an inference from their architecture. It is in their own published guidance: score a low single-digit to low double-digit percentage of traffic at volume; sample above a set number of prompts; sample a fixed number of traces per report.

So the honest statement of the difference is tighter, and much harder to argue with.

> Their production experience signal is judged and sampled. It has to be, because a well-defined rubric is the only thing their deterministic layer can compute over, and rubrics do not survive contact with production.

That is disqualifying, for reasons that are not about model capability and will not be fixed by a better model.

- **It is non-deterministic, and the non-determinism sits in the judge.** The same interaction submitted twice can score differently. The model's own randomness, provider-side changes, silent model updates, sensitivity to how the rubric is worded: all of it moves your number while nothing about your agent moved. Setting the temperature to zero narrows the first of those and touches none of the rest. That is disqualifying, and not as a matter of opinion. The entire purpose of a metric is spotting a difference: is this release better than the last one, is this cohort worse than baseline, is the trend improving. When the instrument has its own variance, you cannot separate a real regression from measurement drift. You end up debating the number instead of fixing the agent.

- **It cannot be reproduced or audited.** Re-run last quarter's data through today's judge and you will not get last quarter's answer, because the model has changed and possibly been deprecated. Every historical baseline is invalidated by an upgrade you did not control and may not be told about.

A measurement you cannot re-derive is not evidence. It cannot be defended to a finance organization, held up in a regulated review, or used to settle a disagreement between two teams, which is most of what an operating metric is actually for. In video, the Quality of Experience (QoE) definitions had to stay stable for a decade before anyone would manage a business against them. Stability is a feature of the definition, not a nice-to-have.

- **Its biases are systematic, not random.** Judge models have well-documented and consistent preferences: for longer answers, for fluent and confident phrasing, for outputs resembling their own family's style, for whichever option was presented first. Random error averages out across a population. Systematic bias does not; it accumulates in the same direction at every scale. And the bias here is correlated with exactly what is being measured: a verbose, confident, ultimately unhelpful agent is the specific failure mode a judge is most likely to score well. You are not adding noise, you are indexing in the direction of your worst blind spot.

- **The cost curve is the wrong shape.** You are spending inference to evaluate inference, at roughly the same unit economics as the workload itself. Cost rises in step with traffic and never gets cheaper per unit, which means the more you measure the less you can afford to measure. The published guidance from tools built this way is to score a low single-digit to low double-digit percentage of traffic at volume, not because sampling is methodologically sound, but because the arithmetic forces it. A measurement system should get cheaper per unit as you scale it. This one competes with production for the same budget line.

- **It is too slow to be in the real-time path.** A judge call costs about what the agent call cost. Applied across a population, measurement lands well behind the interaction it describes, which puts you in postmortem territory precisely when the window to save the customer was open.

- **It is still capped by the rubric.** A judge can only assess what someone wrote down. That is evaluation with extra steps: bounded by what you anticipated, blind to what you didn't. The failure you most need to find is the one no rubric author imagined, and no amount of grading finds it.

- **It cannot do population math.** Ask a judge which cohort is drifting from baseline, or whether a difference between two segments is real or noise, and the question does not even make sense. That is a query over a dataset, not a judgment about a document. Statistical investigation and case review are different activities, and the market keeps substituting the second for the first because the second is easier to build. You cannot read your way to the truth at a million conversations a day, not with people and not with models reading on your behalf.

Stack these and they compound in the same direction: a sampled subset, scored by a biased instrument, at a cost that prevents you from widening the sample, too late to act on, bounded by what you already knew to look for, and impossible to reproduce next quarter. Each defect makes the others harder to detect.

The alternative is not a better judge

The usual response to all of this is to improve the judge: a stronger model, a tighter rubric, an ensemble, a panel, a judge that grades the judge. That is the wrong correction. The problem is not the quality of the judge. It is that the judge is in the wrong seat.

There is already a judge in every one of these conversations, and it is not a model. It is the party being served. They pass judgment constantly, without meaning to: by repeating themselves, by rephrasing, by quietly giving up on part of what they asked for, by opening a search tab while the conversation is still going, by leaving to verify the answer, by escalating to a human, by finishing the purchase somewhere else, by not coming back. An agent consumer does the same thing in its own vocabulary: retry, reformulate, discard, fall back, fail downstream. None of that is an opinion about the agent. It is what happened, and it settles the question in a way no score does.

User-as-a-judge, then, rather than LLM-as-a-judge. The measurement question stops being how would a model grade this response and becomes what did the party on the other end actually do about it. That question has a determinate answer. It does not drift when a provider ships a new model, it does not need a rubric written in advance, and it cannot be flattered by a verbose, confident, unhelpful reply, which is the specific failure a judge model is most likely to score well.

It does require reading both layers at once and joining them: the signals inside the trace that show what the interaction cost the person, and the signals outside it that show what they did as a result. Neither half is sufficient on its own: inside the trace alone you see effort without consequence, and outside it alone you see consequence without cause. Joined, on the same interaction, they are the closest thing to ground truth this market has.

Which is precisely why this is an architecture question rather than a modeling one. Reading a verdict that is written in behavior, across a whole interaction, on every interaction, while it is still open, is a computation. None of this makes language models bad. It makes them the wrong component for this position in the architecture. Section 4 is about which position they belong in.

SECTION FOUR

Deterministic compute at scale is the foundation

Start from what experience measurement actually examines: a conversation. A sequence of exchanges between a person and an agent, in order, with timing.

That sequence is a data structure, and what happened inside it is not a matter of opinion. The person put the same request in again in different words. The agent went back over ground it had already covered. The person let go of part of what they originally asked for. The request had to be reformulated before the agent engaged with the right thing. The exchange kept going without converging. The same tool was called repeatedly with the same arguments. The person stopped mid-conversation, went elsewhere, and came back.

None of that requires a judgment. You do not need to ask a model whether someone repeated themselves; you need to have held the conversation's state and compared. Each of those is an event in a sequence: detectable by rule, countable, and identical every time you compute it.

> **That is the reframe. Experience measurement is computation over the structure of a conversation. Counting, ordering, timing, comparing. It is instrumentation, and it is exact.**

The surrounding surface belongs in this too, in its correct proportion. The second layer of experience, what the person did during and after, is a specific, bounded set of actions attached to the conversation record: they used the search bar while the conversation was still open, they corrected something afterward, they went and verified it, they escalated to a human, they left and didn't return. Those get joined on to the conversation and computed the same way, as additional facts about that conversation.

Four requirements have to hold at once, and Section 6 sets out the full standard they belong to:

Stateful, because the unit is a whole conversation, carried and updated as it unfolds, across turns and across agents. A repetition, a reformulation, a loop, a dropped handoff, a failure to converge: none of those exist in a single turn or a single agent's span. They exist only in relation to what came before, and often in relation to what a different agent did before. A pipeline that scores turns independently has discarded the structure the metric lives in.

Highly-dimensional, because every conversation arrives with its own context: what kind of request it was, how it was phrased, which agents participated and in what order, which tools were invoked with which parameters, where in the sequence it went wrong, which surface it happened on, what each agent was configured to do at the time. Failures concentrate in specific combinations of those, so isolating a cause means crossing them, which means carrying all of them, on every conversation.

Full-population, because the failing case is typically a narrow class of conversations that the average conceals: one kind of request, one parameter omitted from one call, one handoff between two specific agents. Sampling is precisely what hides it.

Real-time, because a problem you find tomorrow is a conversation you already lost, and the most valuable version of this acts while the conversation is still open.

Any two or three of those are doable. All four at once, at production volume, is where architectures fail; they will give you stateless and fast, or stateful and batch, or dimensional and sampled. That combination is the real barrier to entry in this category, and it is why the gap has stayed open while everyone agrees it exists.

It is also what makes the multi-agent failure problem from Section 2 something you can actually fix, rather than just name. Measuring end to end and then resolving to the responsible agent and turn requires holding the whole interaction as one stateful object, carrying every participant as a dimension, and doing it on all traffic, which is the same list. There is no lighter-weight architecture that gets you multi-agent diagnosis. It falls out of this one or it does not happen.

The economics also invert relative to the judged approach. Deterministic compute carries real fixed cost (this is years of platform engineering, not a weekend prompt), but the marginal cost of measuring one more conversation trends toward nothing. That is what makes full census affordable, which is what makes the narrow failing class visible at all. The judged approach has almost no fixed cost and punitive marginal cost, which is why it always ends in sampling.

> **The choice between architectures is a choice about whether you can afford to see every conversation.**

Determinism buys a second thing that is easy to undervalue until you need it: a stable definition. The same input produces the same value this quarter, next quarter, and after your model vendor deprecates something. That is what makes trends real, comparisons fair, and disagreements resolvable. It is what allows a number to become the number a business is run on.

So where does AI belong? Everywhere except the measurement path, and the work it does there is essential.

The genuinely hard part here is semantic: reading language to establish what a person was actually asking for, in their own words, and what kind of request it was. Counting cannot do that. So the correct pattern is to use models to label, then compute deterministically over the labels. The model reads a turn and labels it; that label is written onto the conversation as a dimension; the metric is then computed exactly over the labeled sequence. That distinction is the whole architectural argument in one line:

> **The model's judgment is an input to the measurement, never the measurement itself.**

Above the deterministic measurement layer, AI earns its place again by explaining why one class of conversation is drifting from its peers, proposing where in the harness the cause probably sits, naming which agent in the chain is responsible, summarizing what the failing conversations have in common, and letting an operator interrogate all of it in plain language instead of writing queries. That is an analyst sitting on top of exact numbers, which is the right job for a probabilistic system, meaning one that can be wrong, because its output is a theory a human checks rather than a fact the business is run on.

Which brings up the point most often missed in this market: the quality of an analytics agent is set almost entirely by the platform underneath it. Everyone is adding an agent; the ones that will be worth anything are sitting on a layer that computes conversation structure statefully, dimensionally, completely, and live. The model commoditizes fast. The platform underneath does not.

SECTION FIVE

What “experience” actually refers to: two layers, routinely conflated and partially delivered

Sections 3 and 4 argued about how to measure experience. This section and the next one say what is being measured, and to what standard. The term needs pinning down first, because the market currently uses “experience measurement” to mean one of two quite different things, and treats whichever one it means as the whole.

The first layer is inside the interaction: the efficiency of the agent’s own work. How much effort the interaction demanded of the person: how much repetition, restatement, backtracking, compromise, and patience it took to arrive somewhere. This layer is derived from the trace, and the trace is the right place to derive it from: repetition, reformulation, backtracking and stalling are all events in the record of the conversation. It requires no external instrumentation. But note what it is not. It is not a measure of response quality: every individual response in a long, grinding, expensive interaction can be excellent. Excellence per turn does not aggregate into a good experience. Effort is a property of the whole conversation.

The second layer is outside the interaction: what the party being served reveals through their own behavior, during the conversation and after it. What they did next, what they went around the agent to do themselves, what they came back to verify, whether they escalated, whether they finished the task somewhere else, whether they returned at all. This is what they actually did, rather than what they said, and it is the closest thing to the truth available.

This holds whether the party on the other end is a person or another agent. Increasingly it is another agent, in B2C and in B2B alike, and the framing does not need adjusting to accommodate that. An agent consumer arrives with an intent, stated as an explicit task or an output contract, and it reveals its judgment through behavior exactly as a person does: it retried, it reformulated the request, it discarded the output, it fell back to another path, it failed downstream. Different vocabulary, identical signal, same measurement.

Most commentary collapses these. Vendors with behavioral data argue the second layer is experience measurement; vendors with trace data argue the first is. Both positions are self-serving and both are incomplete, because each layer catches precisely what the other misses.

> Inside the trace alone, you see effort without consequence. Outside it alone, you see consequence without cause.

An interaction can be efficient and still have failed: brief, clean, and followed by the customer quietly doing it themselves. An interaction can be a grind and still reach the outcome; the customer was already committed and pushed through, which registers as success while you spend down goodwill you will not get back.

Which is where the trace vendors sit, and it is worth being exact about how far short they fall. An experience layer built only on traces is partial by construction. The trace stops when the conversation stops, and everything the person did next happened somewhere the trace cannot see. So the first layer, on its own, is half an instrument. That is a limit of the data, not a criticism of anyone.

If there is a criticism, it is this: trace vendors do not deliver that half either. What they compute from the trace is quality of response. Was this answer accurate, on-brand, defensible. That is narrower again, in two ways that compound. It takes the agent's output as its subject rather than the person's effort, so it is not measuring experience at all. And it resolves per turn rather than across the interaction, so even the part it does measure has the sequence stripped out of it. They are not delivering less experience data. **They are delivering a part of a partial layer, and reporting it as the whole.**

A second group gets further than that, and they deserve to be separated from the first. These vendors read the whole trace rather than scoring turns in isolation: the full sequence of one conversation, in order, with the tool calls and the handoffs in view. That is a real improvement, and it recovers the structure the per-turn scorers throw away. It still stops one step short of experience intelligence, and the reason is not effort or maturity. A single conversation cannot tell you whether what you are looking at is unusual.

Nine turns is a warning sign or a healthy signal depending entirely on what nine turns means for requests of that kind, from customers like that, on that surface, at that moment. The conversation does not contain that answer. It lives in the population, and it only becomes visible when you carry enough dimensionality on every conversation to cross one piece of context against another until the failing segment separates from the rest. Read one conversation perfectly and you have an anecdote. Read all of them, dimensionally, and you have a cause. You cannot read your way across that gap at a million conversations a day, not with people and not with a better viewer.

That last case is the one the market underrates most significantly. Reaching the outcome and losing the customer are entirely compatible events. Outcome is not experience, and any measurement practice that treats a completed transaction as proof of a good one is measuring the wrong thing confidently.

Joining the two layers is the whole game, and it is worth being precise about why that is hard rather than merely unusual: both halves have to be attached to the same interaction, at the same time, on every interaction, or the join is a research exercise rather than an instrument. The trace gets you one half, computed properly rather than scored. The surrounding behavior gets you the other. And neither half means anything until it is read against the population, which is where a value stops being a number and starts being a finding. Those are three different capabilities, and a category built on the trace alone has a ceiling it cannot engineer past. Hold onto that. This is the point Sections 3 and 4 grounded on.

SECTION SIX

What an experience metric has to be

If both layers are in scope, the question becomes what qualifies as a metric worth running an operation on. This is where the market lacks a shared standard, and where a standard is worth more than another dashboard. The six requirements are:

1. **It must take the served party as its subject.** Not the component, not the model output, not the response, but the party the agent was working on behalf of. Usually that is a person. Increasingly it is another agent, and the requirement does not weaken: an agent consumer has an intent and reveals its judgment through behavior, as Section 5 describes, so the subject of measurement is the same and only the vocabulary changes. The test holds in both cases. If the metric would read the same value whether or not anyone was served on the other end, it is a system metric wearing a new label.

2. **It must be scoped to the whole interaction.** Experience is cumulative and sequential. It accrues across turns, and across agents, and it depends on order and timing. A metric computed per turn and averaged has destroyed the only structure that carried the meaning. A metric computed per agent has destroyed it twice.

3. **It must be computed, not judged.** Deterministic, reproducible, auditable; the same input producing the same value every time. This is the requirement the market is furthest from, and it is the subject of Sections 3 and 4.

4. **It must hold at full population.** A metric you can only afford on a slice of traffic is a research finding, not an operating instrument. Full census is not a purity argument; it is the only way the decisive cases stay visible. The failure you most need to find usually lives in a narrow class of conversations that the average conceals, and sampling removes it by design.

5. **It must arrive while the outcome can still be changed.** Measurement that lands tomorrow describes a customer you already lost. Real time is not a performance feature here; it is what separates an operating metric from a postmortem.

6. **It must resolve to a cause, which means it must be read against the right baseline.** A number that tells you something is wrong without telling you where is an alarm, not a diagnostic. Getting to a cause requires two things: enough dimensionality to cross one piece of context against another until the failing segment shows itself, and comparison against a baseline rather than a fixed threshold, because the same value legitimately means opposite things for different populations. A long, effortful interaction is a warning sign for a customer who wanted a routine task done and a perfectly healthy signal for one making a considered, high-consequence decision. Absolute thresholds erase that distinction. Contrast preserves it.

Every vendor in this market will tell you they compare against a baseline. So asking a vendor whether they compare against a baseline tells a buyer nothing. They all say yes, and they are all right. The question that separates them sits one level down. **What is the baseline made of?**

The eval camp's baseline is historical or curated: a prior experiment run over a fixed matched dataset, or a golden reference set. Both answer one question, "Is this worse than it was?," and answer it well. That is the correct question for a release gate and the wrong one for a production operation, because it cannot see the segment that has been failing since the day it shipped. There was no better "before" to regress from. To a historical baseline, a chronically broken class of conversation simply is the baseline. And a curated set contains, by construction, only the cases someone thought to curate.

The baseline experience measurement requires is contextual and peer-relative: this slice of live traffic, against what is normal for conversations of its kind, right now, in the same population. That is the comparison that finds the segment that never worked well, as opposed to the one that recently got worse. Conversations that were never right are the majority of what is actually wrong with production agents, and no golden dataset can be built to find them.

SECTION SEVEN

What those metrics have to tell you

Requirements and architecture describe how the measurement gets built. None of that matters unless the numbers answer questions someone can act on. Three, in order:

Did the person get what they came for?

Not whether the system completed, not whether the response was defensible, not whether the case was marked resolved. Whether the end user on the other end obtained the thing they arrived wanting.

What did it cost them to get there?

Effort, patience, time, repetition, compromise. This is the question the market currently has almost no language for, and it is where most agent programs are quietly failing: not in outright breakage, but in interactions that succeed expensively.

What did that do to their willingness to come back?

The question that matters most, and the one that converts experience measurement from a quality exercise into a financial one.

There is a fourth, addressed to the operator rather than the customer: **what specifically do I change, and who owns it?** A measurement practice that ends in a score has produced a scoreboard. One that ends in a named cause, this agent, at this turn, on this class of request, because of this instruction or tool definition or default or missing piece of context, has produced work. In a multi-agent system that second half is not a nicety; without it the finding lands on five teams at once and moves nowhere. Nearly all of it is fixable in the surrounding harness rather than the model, in hours rather than months, at negligible cost. But only by a team that can see which situation is failing, and which participant caused it. That visibility is the entire bottleneck. And you are paying dearly for it, every time it happens.

SECTION EIGHT

The largest line item in an AI program is bad experience

Most of this paper has argued that experience measurement is a quality discipline. It is also, and this is underappreciated to the point of being a market blind spot, the largest cost lever in a production agent program.

Not a co-benefit. The largest one.

Start with the arithmetic, because it is not intuitive. Cost does not scale linearly with the length of a conversation. Every turn resends the accumulated context: the first exchange processes a small input, the fifth processes everything that came before it plus the new message, the ninth processes all of that again. Total consumption across a conversation therefore grows with roughly the square of the turn count, not in proportion to it. A conversation that takes nine turns instead of three does not cost three times as much. It costs closer to an order of magnitude more.

Prompt caching and context management change the multiplier. They do not change the shape. The cached context is cheaper per token and it is still billed, and it still grows with every turn.

Then add the loop. A single conversational turn is frequently not one inference. An agent that reasons, invokes a tool, observes the result, reasons again, and invokes another tool has run several round trips to produce one reply. In a multi-agent system that one reply may have passed through an orchestrator and two specialists, each carrying its own context window. On a reasoning model it has also generated thinking tokens that the customer never sees and you certainly pay for. So a wasted turn is not a marginal unit of spend. It is the most expensive kind of unit there is.

Now compare the levers teams are actually pulling. Switching to a cheaper model moves unit price by a bounded percentage. Trimming the system prompt shaves the base context once. Caching and batching are real multipliers on an unchanged volume of work. All three optimize the price of the work being done.

Only one lever reduces the amount of work: whether the agent understood the person and got them there without making them repeat, restate, backtrack, or settle. That variable has no ceiling, it grows faster than the turn count, and it is set entirely by experience quality. It dwarfs the others, and almost nobody is managing it as a cost input.

The reason it stays invisible is the part that matters most. Every one of those wasted turns is a well-formed response. It passed its eval. It met its latency target. It threw no error. It does not appear anywhere as a failure. It appears as usage.

Which produces something close to a backwards incentive: an agent that is failing its customers generates traffic that reads as engagement and bills as success. Tokens per conversation trending up gets interpreted as depth of adoption. It is at least as likely to be friction, and no dashboard in the standard stack can tell the difference.

> Your inference bill is, in substantial part, an itemized invoice for your agent's misunderstandings. Almost no one reads it that way.

The worst case is worse still. A customer who abandons mid-conversation was billed for every turn up to the moment they gave up, and produced nothing. Not a resolution, not a sale, not a deflected ticket. That is pure spend against a negative outcome, and it is the cell in the grid where experience failure and cost failure are the same event.

Which brings the argument back to measurement, because none of this is actionable without it. Aggregate token spend tells you the total. It cannot tell you which conversations ran long because the underlying task was genuinely complex and which ran long because the agent did not understand what was being asked. Those are opposite situations that look identical on a usage report, and the correct response to each is the reverse of the other. Separating them requires the same instrument this paper has been describing: conversation-level, computed, full-population, dimensional. Without it, cost reduction in an agent program turns into blunt moves like capping turns or downgrading models, both of which make the experience worse and can raise total cost.

With it, something becomes available that traditional software never offered, and the novelty of it is worth naming rather than assuming.

In conventional enterprise software the two goals were structurally opposed. A better customer experience meant more infrastructure, more features to build and maintain, more people staffing support. Cost reduction meant fewer of each. That tension sat underneath most enterprise technology decisions of the last three decades, which is why experience investment has always had to be argued as a revenue case against a cost objection: the finance owner and the experience owner were competing for the same budget because they genuinely wanted opposite things. Agent operations inverts that relationship. The friction and the waste are not two problems in tension, they are one event counted twice. Reducing the effort a conversation demands of a customer reduces what that conversation costs to serve, in the same motion, with no offsetting spend. The market has not restructured around this yet. Most organizations still run agent cost and agent performance as separate programs, under separate owners, with separate targets and a standing argument between them. That arrangement is inherited from a constraint the technology has removed, and the companies that notice first will spend less and deliver better at the same time while their competitors are still negotiating the tradeoff.

SECTION NINE

A hypothesis: experience is the missing vector in model selection

Everything above concerns whether an agent works. This section advances a further claim, offered as a hypothesis rather than a settled finding, because it is where I think the financial case for experience measurement becomes impossible to argue with.

The premise: multi-model architectures should be the norm, and the industry already half-knows it. Model performance is neither consistent nor predictable. Capability arrives unevenly. One model is materially better at structured tool selection, another at long multi-step reasoning, another at concise conversational handling, another at a specific language or domain, and the ordering reshuffles with every release. Nobody is best at everything, the leaderboard has a shelf life measured in weeks, and public benchmarks

measure performance on someone else's tasks with someone else's prompts and tools. Committing an entire agent estate to one model is therefore a bet against a distribution you can observe changing in real time.

So routing across models is the rational architecture. The question nobody has answered well is: **routing on what?**

Today the answer is cost, because cost is the only number anyone has. Price per token is published to four decimal places. Latency is measurable in milliseconds. The experience delta between two models on your specific traffic is, for most organizations, entirely unknown. And an arbitrage where you can price one side of the trade and not the other is not arbitrage. It is a guess with a spreadsheet attached, and it will systematically optimize the side you can see, which is exactly backwards.

The hypothesis is that experience is the first vector and cost is the second, in that order. Route each class of request to whichever model demonstrably produces the better experience for that class in production. Then, wherever two models are at measured experience parity, take the cheaper one, every time, without hesitation, and revisit it on every release.

That ordering is not a philosophical preference. It follows from arithmetic, and here is the financial argument in five parts.

1. **First: the frontier default is an unpriced insurance premium.** Most teams route nearly everything to the strongest model they can justify, because they cannot prove a cheaper one is adequate. That is a rational response to missing information and an expensive one: you are paying a premium on every conversation to hedge an uncertainty you have chosen not to measure. Measurement converts that premium into a decision. Every request class where a cheaper model holds parity is margin that was already yours, recovered at zero experience cost. There is no cleaner line item in an AI budget.

2. **Second, and more important: unit price is not cost.** This is where cost-led routing destroys value while appearing to create it. A cheaper model that handles a request class slightly worse produces more restatement, more clarification, more backtracking, more tool invocation, and every one of those is billable. Take the arithmetic on its face: a model priced 40% below the incumbent that requires meaningfully more exchanges to reach the same place can be more expensive per completed conversation than the model it replaced, while also delivering a worse experience. You will have paid more for less and booked it as a saving, because the savings appear in a unit-cost line and the loss appears in volume, latency, and churn.

The correct denominator is **cost per successfully completed conversation,** not cost per token or cost per call. That number cannot be computed without experience measurement. Which means the routing decision most enterprises are making today is not merely under-informed; it is being made on a metric that can move in the opposite direction from the thing it is meant to control.

3. **Third: the arbitrage opportunity refreshes continuously, and only measurement lets you act on it.** Inference cost at constant capability has been falling roughly an order of magnitude a year. Every few weeks a cheaper model becomes plausibly good enough for some slice of your traffic. Without a production experience signal, each of those releases is a risk you defer, because nobody wants to be the person who downgraded the customer-facing agent to save a few points of margin. With one, each release becomes a chance to re-price your traffic, and you run it in days: test on live traffic, measure experience per request class, move what holds parity, leave what doesn't. That is a compounding advantage, and the compounding runs against competitors who are still deferring.

4. **Fourth: the unit of arbitrage is the request class, and in a multi-agent system it is the request class per role.** You do not choose a model. You choose a model per situation, and situations are where models diverge most sharply, and in a chain of agents the orchestrator, the retrieval step, the domain specialist, and the summarizer have genuinely different requirements. This is the point at which the dimensional requirement from Section 6 stops being a technical nicety and becomes the enabling condition for the entire financial strategy: capturing arbitrage requires crossing model against request type against agent role against tool path, on full population, and reading which combination wins where. A single blended quality score across all traffic tells you nothing actionable, because the whole opportunity lives in the variance the average destroys.

5. **Fifth: single-model dependency is concentration risk, and it is priced as if it were free.** One vendor sets your unit economics, your capability ceiling, your deprecation schedule, and your rate limits at your busiest moment. Diversification is the standard answer to concentration risk in every other part of a business. But a portfolio you cannot measure is not a portfolio; it is an assortment. Experience measurement per request class is what turns a set of model integrations into an actual allocation decision, with the same logic any treasurer would recognize: know the return on each position, rebalance when the pricing moves.

The conclusion, and the part worth taking to a CFO: an experience measurement layer is not only a quality investment. It is the decision function for the largest controllable line item in an AI program. The engineering work of multi-model routing is largely solved: gateways, abstraction layers, and standard tool protocols have commoditized the plumbing. What is missing is the evidence to route on. Supply that, and the layer is funded out of the arbitrage it enables: cheaper models wherever experience is at parity, stronger models only where they measurably earn their price, and no more paying frontier rates as a hedge against your own blind spot.

The experience layer stops being a cost center at that moment. It becomes the instrument that prices the rest of the stack.

SECTION TEN

Market implications

The incumbent layers become table stakes, not differentiators. Tracing and evaluation are not displaced; they are conceded and absorbed. Every serious agent operation will have them, and having them will distinguish no one. Value migrates up, to whoever can prove what the interaction did to the customer.

Architecture, not features, will decide this market. This is the practical consequence of Sections 3 and 4, and it is the part a buyer should weigh hardest. Any vendor can add a metric in a quarter. No vendor can make its metrics deterministic, complete and real-time unless the platform underneath was built for it, and platforms are not retrofitted. So a feature comparison will tell a buyer very little.

Seven questions will tell them a great deal, because each one has a factual answer:

1. What fraction of production traffic is measured, all of it or a sample?
2. Is the experience signal computed, or judged by a model?
3. How fast does it land?
4. Can last quarter's number be reproduced today?
5. What is captured outside the trace?
6. What is the baseline made of: a prior run, a curated set, or live peer traffic?
7. Where a model is part of the solution, what is its measured accuracy?

That last question is the most revealing, because most vendors cannot answer it. A model that sorts conversations into kinds can be checked against what those conversations really were, and that produces an accuracy figure. A model that hands out scores cannot, because there is no right answer to check it against. If a vendor has no figure to give you, that is the answer.

Multi-agent is where the gap widens, not narrows. Every vendor in this market is getting better at the single trace. Meanwhile the systems being deployed are getting more layered, which moves the failures further from anything a per-span score can represent. A category that measures parts is walking away from where production is going.

The stakeholder changes. Evaluation tooling is bought by platform and engineering teams solving for safe release. Experience measurement is bought by whoever is accountable for whether the agent works, increasingly the CTO, with AI leadership, product, and digital experience alongside. That is a different conversation, a different investment, and a materially different set of consequences for getting it wrong.

Cost and quality stop being a tradeoff. This is the most commercially useful implication and the least discussed, and Sections 8 and 9 are the two halves of it: within a given model, poor experience is the dominant driver of consumption, and across models, experience is the vector that should govern routing. Both point the same way. Very few enterprise investments improve the customer's experience and reduce the bill through the identical intervention, and agent operations should be argued on that far more aggressively than it currently is.

The window is short. The incumbent categories are moving in this direction: sentiment signals, frustration templates, usage analytics, sampled discovery features. They are walking toward the same ground from the wrong architecture, which is a real competitive dynamic and not a reason for comfort. The definitional advantage goes to whoever adopts the quality of experience standard clearly while the market is still deciding what the words mean.

SECTION ELEVEN

The precedent, and why it is predictive

Every claim in that last section is a prediction, and a prediction is worth whatever the precedent behind it is worth. This one has a precedent. The transition has already run once.

Video streaming began with delivery metrics. Did the stream start, what was the bitrate, what was the error rate. The whole industry ran on it, and it was genuinely necessary. It was also insufficient in a way that took years to become consensus: services watched clean delivery dashboards while subscribers churned, because a stream that played was never the same as a session worth watching.

What changed the industry was a shift in the subject of the measurement, from the delivery system to the viewer, measured on every session rather than a sample, in real time rather than in a report, with enough dimensionality to isolate which device, network, geography, or title was actually responsible. Quality of experience went from a differentiating idea, to table stakes, to something you could not run a business without. The companies that made the shift early set the operating standard for everyone else.

Worth noting what made it possible: none of it could have been done by inspection. The industry did not get there by having experts watch streams. It got there by building deterministic real-time computation over every session, then arguing about what the numbers meant. The measurement infrastructure preceded the consensus, and it is the reason the consensus could form at all.

The analogy holds because the failure mode is identical: a technically functioning system, a dissatisfied customer, and no instrument connecting the two. It holds more strongly, in fact, because a conversation carries far more signal about a person's intent and satisfaction than a video session ever did, and because the addressable market for agent interactions is a large multiple of the one for streaming.

Where the analogy needs care: agent interactions are about language in a way video sessions are not, and they are built from many parts in a way video sessions never were. Establishing what someone actually wanted requires language understanding that has no equivalent in delivery analytics, and attributing a failure to one participant in a chain of agents has no equivalent either. Both are real. The first is precisely the work described in Section 4: models labeling situations and feeding a deterministic pipeline, rather than models being asked to be the pipeline. The second is what the stateful, dimensional, full-population platform exists to make possible.

What decided that market in the end was not who saw furthest. It was who adopted a quality of experience operations focus, while others tried to troubleshoot system alerts, and therefore held the only viewer evidence in the market, while others were blind. That is the whole lesson, and it is on offer again.

SECTION TWELVE

Where Conviva Stands

We are the ones who built that solution. Conviva pioneered quality of experience, moving the industry off delivery metrics and onto the viewer, on every session, in real time, with the dimensional depth to isolate a cause rather than raise an alarm. Doing that at live-event scale left no choice but to build the platform described in Section 4: stateful, high-dimensional, full-population, and real-time simultaneously, computed deterministically. That platform is the asset. It took years, it is the hard part, and it is why the agent problem looks familiar to us rather than novel.

Conviva applies that discipline to AI agents. We measure both layers of experience, inside the trace and outside it, joined on the same interaction, on the full population, in production, and contrastively against live peer traffic, so that a failing situation isolates and resolves to a specific agent, a specific turn, and something a team can change today. AI does the semantic work: reading language to establish what a person actually came for, naming patterns, explaining deviations, answering questions in plain language. The measurement itself is computed. We call the methodology EPIC:

E	P	I	C
EXPERIENCE OVER RESPONSE	POPULATION OVER SAMPLE	IN PRODUCTION OVER IN THE LAB	CONTRASTIVE OVER ISOLATED

> The position, plainly stated: your evals grade the agent; we let the user grade it. Passing an evaluation is necessary but it is not proof. The proof is in what the person did, and proof requires a solution you can trust twice.

If you are putting an agent in front of your users or customers, you need to be able to prove it worked for them. That capability is not optional infrastructure. The market has not fully arrived at that conclusion yet. It will.