The Crystal Ball That Never Was

Why Observability and AIOps Vendors Keep Missing Their Own Promises

Paul Bevan ©Bloor Research August 2026

ARTICLE REPRINT

I have been analysing and commenting on the whole IT Operations and Service Management (ITOM/ITSM) markets for over a decade, and in that time the Observability and AIOps industry has sold the same dream: feed us your logs, metrics, and traces, and we'll tell you what's about to break before it does – then fix it ourselves. Splunk promised it. New Relic promised it. Dynatrace, Datadog, LogicMonitor, BigPanda, IBM Instana – all of them, in one form or another, have pitched predictive incident detection and automated remediation as the inevitable next step. And yet, year after year, on-call engineers are still getting paged at 3 a.m. for problems the platform swore it would catch, and are still manually running the same runbook the platform swore it would execute on its own.

This isn't a story about bad technology. Individual vendor capabilities vary and some teams do report genuine predictability and automated remediation gains from their Observability and AIOps tools. It's actually a story about a category that, on the whole, has been selling the destination while consistently underestimating the terrain.

The gap between "anomaly detection" and "prediction"

Most of what gets marketed as AI-driven prediction is, underneath, anomaly detection with a better UI. A model learns what "normal" looks like for a metric, flags deviations, and calls it a prediction. That's useful – it can reduce detection time significantly – but it isn't prediction in any meaningful sense. Actual prediction requires understanding causal structure, for example, this queue backs up because that downstream service throttles, which cascades into that connection pool clogging it up, which eventually surfaces as user-facing latency forty minutes later. Vendors have been much better at pattern-matching statistical outliers than at modelling the causal graph of a live, constantly changing distributed system. The industry's own trend pieces keep reaching for the same reassuring language – "moving from proactive to predictive" – practically every year, which is itself a tell: if the shift were actually happening, vendors would stop having to promise it.

Alert fatigue got automated too

The first generation of AIOps was sold as a cure for alert fatigue: correlate the noise, suppress the redundant pages, surface the one alert that matters. In practice, many teams report the opposite experience – a new layer of "intelligent" alerting sitting on top of the old noisy layer, occasionally correlating things incorrectly and eroding trust faster than the raw alerts did. Data from the industry itself is somewhat contradictory and is not always flattering. While every vendors' websites have case studies that show significant reductions in alert noise and mean time to resolution (MTTR) various customer surveys tell a different story. For example, PagerDuty's 2024 State of Digital Operations survey of 500 IT leaders found that customer-facing incidents had risen 43% year-over-year, with the average incident now taking nearly three hours (175 minutes) to resolve – even as AI-assisted observability and AIOps tooling was being marketed aggressively during that exact window, and even as respondents' own IT budgets for automation were increasing. If the tools were delivering on the core promise, incident volume and resolution time should be moving the other way, not climbing alongside AI adoption.

Remediation vendors keep rediscovering the same wall

"Automated remediation" sounds like a finished product in a slide deck and turns into a liability discussion in a change-advisory board meeting. The technical piece – detect a known failure signature, execute a runbook, restart a pod, roll back a deploy – has existed in some form since long before anyone called it AIOps. The hard part was never writing the automation; it was trusting a black-box model to decide when to pull that

trigger on production infrastructure it doesn't fully understand, with a blast radius the model has no real concept of. Even vendors pitching generative-AI-assisted remediation are, by their own description, still framing the AI's job as recommending actions for a human to approve and execute – the fine print, that full autonomy still isn't trusted by the people selling it.

Why the gap persists

Here are a few structural reasons this problem has proven so durable:

- **The data was never clean enough to model causality.** Observability data is fragmented across logs, metrics, and traces collected by different agents, at different resolutions, often missing the topology metadata that would let a model reason about dependencies rather than just correlations.
- **Every environment is a snowflake.** A model trained on "what normal looks like" for one organisation's Kubernetes cluster doesn't transfer cleanly to the next. Vendors sell horizontal platforms; the causal knowledge that actually predicts failure is deeply local and constantly changing as architectures evolve.
- **Incentives favour shipping "AI-powered" features over shipping trust.** Once a vendor can put "predictive" or "AI-driven remediation" on a feature list and a slide, the commercial pressure to claim the capability arrives well before the capability is reliable enough for engineers to actually delegate decisions to it.
- **Organisations aren't structurally ready either.** Even where the tooling is genuinely decent, most teams don't have the topology mapping, the tested runbooks, or the political appetite to let software touch production without a human in the loop – so the "automation" gets scoped down to advisory dashboards.

What "working" would actually look like

The honest version of this technology, at least for now, is a triage assistant: something that reduces noise, groups related signals and gives an on-call engineer a plausible starting hypothesis faster than they'd get one manually. That's real value – it's just a much smaller promise than "predicts problems and fixes them automatically," and it doesn't sell nearly as well on a keynote stage.

The more interesting question is whether large language models change the calculus going forward. LLM-based agents are, at least, better at reasoning over messy, heterogeneous context – logs, tickets, runbooks, Slack threads – than the narrow statistical models this industry has relied on. That's a genuine architectural shift, not just a rebrand. But the trust problem doesn't go away because the model got more articulate; if anything, a fluent, confident-sounding agent that's wrong about root cause is more dangerous than a rigid rule engine that fails loudly.

Retrieval-Augmented Generation (RAG) alone won't get there – and neither will a bigger model

If the bottleneck is that causal knowledge is "deeply local and constantly changing," the natural next question is what architecture actually captures that – and RAG is the obvious first answer, but an incomplete one.

RAG's appeal is real: point a general-purpose model at an enterprise's runbooks, postmortems, architecture documents, and topology graph, and it can reason over that context without the cost, latency, or governance headache of training anything. It's also more auditable, which matters given the trust problem above – a retrieved passage is a citation an engineer can check; a fine-tuned model's internal weights are not. For the explicit, written-down slice of institutional knowledge, RAG is close to the right default.

The problem is that the causal knowledge that actually predicts failure was mostly never written down in the first place. It lives as a pattern the on-call team recognizes instinctively – this particular latency signature two hops upstream, on this particular cluster, usually means that particular disk is about to fill – and nobody documented it because nobody thought to, or because it changed again three deploys later. RAG can only retrieve knowledge that exists as text somewhere. It has no mechanism for learning the tacit, numerical, time-series pattern language that incident response actually runs on.

That's the argument for enterprise-specific small language models (SLM): something continuously trained or fine-tuned on an

organisation's own telemetry and incident history could, in principle, internalise exactly the tacit patterns RAG can't retrieve. But this isn't a clean substitute either. Most enterprises simply don't generate enough labelled incidents to train a good model from scratch. An SLM trained on last quarter's architecture goes stale the moment the topology changes again, and the retraining pipeline needed to keep it current is itself a serious Machine Learning (ML)-ops commitment most infrastructure teams aren't staffed for. An opaque model that's occasionally right about causality is also a harder sell to an on-call engineer than a system that shows its retrieved evidence.

The more likely answer is that neither approach wins alone, and the piece everyone underrates is the one sitting between them: a continuously updated topology and dependency graph – which services call which other services, which changed most recently, which share infrastructure – derived from the live environment rather than a stale CMDB. RAG over documentation answers "what do we know." A lightweight, frequently-retrained model over telemetry answers "what does this pattern usually mean." The dependency graph is what lets either of those answers get scoped to this incident instead of returning something generic. Vendors chasing "our model" or "our RAG pipeline" as the differentiator are arguably solving the easier two-thirds of the problem; the graph is the unglamorous infrastructure work that makes either of the AI layers locally accurate instead of confidently generic.

The vendors who close that gap will be the ones doing the unglamorous graph work, not the ones with the flashiest model demo. I am the eternal optimist, and I have to believe that not everyone else will be back on stage next year, making the same promise again.

ARTICLE REPRINT

BLOOR RESEARCH

+44 (0)1494 311 460
info@Bloorresearch.com
www.Bloorresearch.com