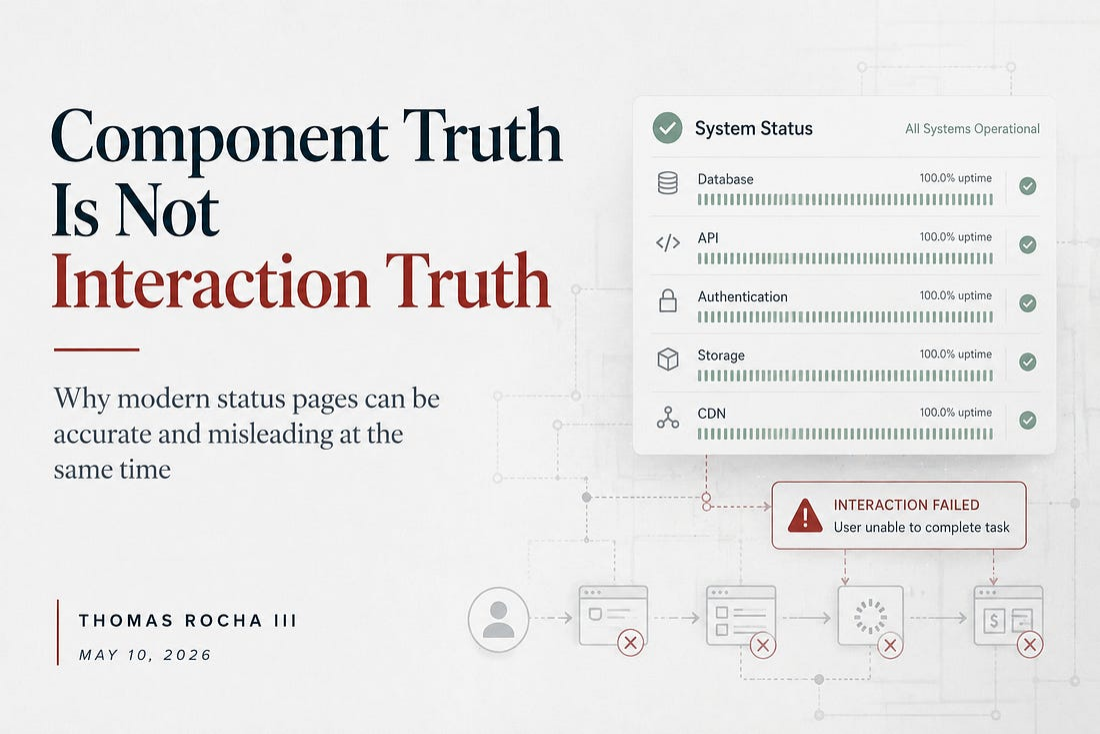
Component Truth Is Not Interaction Truth

Why modern status pages can be accurate and misleading at the same time

THOMAS ROCHA III
MAY 10, 2026



In *Distributed Systems in a Palliative State*, the argument was that the response model has changed without anyone announcing it. The system is no longer being fixed. It is being sustained. Patches ship, incidents close, metrics improve against recalibrated baselines that are lower than they once were. No team is wrong locally. No system is measured as a whole.

This piece is about the public face of that condition.

If the system is in palliative care, the status page is the chart at the foot of the bed. It records component vitals. It does not record whether the patient is still themselves. The two have stopped being the same question, and the public reporting model has not caught up.

The cleanest formulation is this:

> The current status-page model lets vendors hide behind component truth while avoiding interaction truth.

Uptime reports whether pieces are alive. It does not prove the interaction survived.

This is not primarily a transparency problem. It is a boundary problem. Vendors report the boundaries their systems can observe. If the architecture governs databases, APIs, authentication, queues, and services as separate objects, the public report will also decompose every failure into those objects. But the customer does not buy components. The customer buys an interaction. The reporting boundary follows the operational boundary, and the operational boundary follows the architectural one.

That distinction is now visible across every recent major incident. Five examples from the last six months make it legible.

1. GitHub: the platform admits architectural coupling

In March 2026, GitHub published a postmortem identifying its February 2 and February 9 incidents as caused by “rapid load growth, architectural coupling that allowed localized issues to cascade across critical services, and inability of the system to adequately shed load from misbehaving clients.” On April 28, the company followed with a longer statement: it had launched a 10x capacity program in October 2025, then concluded by February 2026 that it needed to design for

30x current scale. The driver, in GitHub's own words, was "a rapid change in how software is being built, especially the acceleration of agentic development workflows since late December 2025."

The independent record is harder. IncidentHub tracked 257 GitHub incidents between May 2025 and April 2026, 48 of them major. February 2026 alone produced 37 incidents. April produced two distinct failure classes in five days: a merge queue correctness defect on April 23 that produced incorrect squash merges across 658 repositories and 2,092 pull requests, and an Elasticsearch overload on April 27 that turned search-backed UI surfaces into empty pages.

GitHub is one of the most operationally mature platforms in the industry. The framing of the public message is "growth pains." The architectural admission underneath it is something larger. A pull request now sits on top of Git storage, mergeability checks, branch protection, Actions, search, notifications, permissions, webhooks, APIs, background jobs, caches, and databases. One slow subsystem distorts several user-facing workflows simultaneously. The component view reports each subsystem's status. The customer view is a workflow that did not complete, or worse, a workflow that completed incorrectly.

The April 23 merge queue incident is the sharper case. The platform reported green. The UI showed checkmarks. The merge results were silently wrong. No traditional uptime metric captures that.

2. Cloudflare: two outages, then "Code Orange: Fail Small"

On November 18, 2025, Cloudflare suffered a global service outage. The trigger was a bug in generation logic for a Bot Management feature file, propagated through the central configuration distribution layer to data centers in three hundred cities. ThousandEyes documented the cascade: organizations using infrastructure providers for content delivery, DDoS protection, bot management, and DNS resolution saw

the impact radiate through layers of dependent services. Email, project management, and CRM tools failed simultaneously for some users. Three seemingly unrelated services with one underlying cause. The common dependency became visible only during the failure.

On December 5, 2025, the network failed to serve traffic for 28 percent of applications behind it for about 25 minutes. The trigger was a configuration change to a security tool deployed urgently to address a React vulnerability.

On February 20, 2026, a subset of customers using Cloudflare's Bring Your Own IP service saw their routes withdrawn via BGP after an automated routing policy configuration error.

Cloudflare's response is more transparent than most: detailed postmortems, declared a "Code Orange: Fail Small" plan, and a stated goal of making the network resilient to errors that could lead to a major outage. But the transparency itself is what reveals the structural condition. Cloudflare's incidents are not failures of redundancy or hardware. They are failures of coordination correctness in a globally coupled control plane. The public incident narrative for each event is a clean summary. The downstream incidents at Cloudflare-dependent services that day, each reported as their own local symptom, never appear in a single coherent public record.

The customer experiences one event. The status pages report many. Cross-vendor coupling of this kind is one of the failure domains the SSOAR work has been mapping.

3. Microsoft 365, January 22, 2026: "infrastructure not processing traffic as expected"

On January 22, 2026, beginning around 14:37 UTC, Microsoft 365 experienced a global disruption that lasted approximately nine hours.

Outlook, Exchange Online, Teams, SharePoint, OneDrive, Microsoft Defender, and Purview were simultaneously affected. Downdetector logged over 30,000 user reports at peak. Microsoft tracked the incident as MO1221364.

Microsoft's public language: "service infrastructure not processing traffic as expected," with mitigation involving traffic rebalancing. Microsoft also noted that an attempted recovery action, a targeted load balancing configuration change, "introduced additional traffic imbalances" that prolonged the outage. Stable mail flow was not achieved until 05:33 UTC the following day. The incident was declared resolved at 18:29 UTC on January 23, nearly 24 hours after the initial disruption. The recovery action that extended the failure is a textbook case of concurrence failure: state changing faster than the system could reconcile it.

The wording mismatch is what matters. "Service infrastructure not processing traffic as expected" describes a routing fact. The customer experience was institutional operational continuity degradation. For organizations that run their internal nervous system on Microsoft 365, the outage was not a Teams disruption or an Outlook disruption. It was a coordination failure in which email, meetings, file access, security tooling, and admin portals all stopped working at the same time, and the recovery effort itself extended the failure.

The component vocabulary cannot represent that. The interaction vocabulary does not exist in the public report.

4. Anthropic: the API is healthy, the model is not

In September 2025, Anthropic published a postmortem describing three separate infrastructure bugs that degraded Claude response quality across August and early September. The disclosure stated plainly: model quality is never reduced due to demand or load; the problems

were infrastructure bugs. The company identified specific affected windows for Claude Sonnet 4 (August 5 to September 4), Claude Haiku 3.5 and Sonnet 4 (August 26 to September 5), and Claude Opus 4.1 (a 56.5-hour window from August 25 to August 28 caused by a botched inference stack rollout).

In April 2026, Anthropic published a second postmortem covering Claude Code degradation between early March and mid-April. Three separate changes had interacted in unexpected ways: a reduction in default reasoning effort from high to medium, a context-clearing routine that contained a bug causing it to repeatedly wipe context, and a shorter-response tweak that affected behavior outside its intended scope. In Anthropic's own words: "Because each change affected a different slice of traffic on a different schedule, the aggregate effect looked like broad, inconsistent degradation."

This is the category that traditional status pages cannot represent at all. The API was operational. Authentication worked. The endpoints returned. What changed was correctness, reasoning depth, instruction-following fidelity, and tool-use behavior. The system was up. The system was also producing degraded output that users noticed before the company acknowledged it.

A status page that reports availability is silent on semantic degradation. The interaction the customer paid for, a workflow producing reliable output, was failing while every component was green. Anthropic's willingness to publish detailed postmortems is the right response. The point is that the underlying disclosure category, semantic correctness during a live agentic interaction, has no analog in the existing operational reporting model. This is the same gap the compliance boundary describes from the regulatory side: vendors cannot report what their architectures cannot govern, and authorities cannot yet require what no one can produce.

The customer was not paying for an API. The customer was paying for a system that reasoned reliably under instruction. Component truth and interaction truth diverge.

5. Retries become the outage

Across 2025 and 2026, the engineering literature has begun to publicly acknowledge a pattern that for years was treated as a tail risk: the recovery logic itself is now frequently the dominant failure amplifier. A November 2025 arXiv paper, *RetryGuard: Preventing Self-Inflicted Retry Storms in Cloud Microservices Applications*, documents how default retry patterns layered atop REST and gRPC transports can turn into self-inflicted Denial-of-Wallet scenarios. A separate November 2025 arXiv submission, *Looking Forward: Challenges and Opportunities in Agentic AI Reliability*, presents an eleven-layer failure stack and notes that errors in one agent can cascade across dependent agents, and can also propagate vertically across layers in ways that monitoring at any single layer cannot detect.

The shift is structural. Historically, failures were the primary event and retries were the mitigation. In the current architecture, retries are routinely the dominant event. AI agents make this materially worse. They retry faster, fork workflows, delegate recursively, accumulate context, and increase coordination load while ostensibly recovering. This is the coordination limit expressing itself in operational form: the cost of maintaining coherence has begun to exceed the work the system is performing, and the recovery logic is where that cost shows up first.

The GitHub incidents from February and April 2026 sit inside this pattern. The April 28 GitHub statement explicitly identifies write amplification, cache rewrite storms, and “retries amplify traffic” as primary contributors to the cascade. The Microsoft 365 January 22 outage extension came from a recovery action that added imbalance.

Cloudflare's incidents propagated through automated control-plane updates whose corrective behavior compounded the trigger.

The implication is that "operational" has stopped meaning what it once meant. A system can be technically online while its recovery behavior is what is bringing it down. The status page reports the trigger, often weakly. It does not report that the recovery logic ate the building.

The boundary chosen for the sentence

Each of these incidents produced public statements that were, in their narrow framing, true.

The API was operational. Authentication was available. The database was healthy. Only a subset of users was affected. No data loss occurred. The service remained online. Infrastructure was restored to a healthy state.

Each of these statements was also a category-level evasion. The customer's workflow could not complete. Authority could not be maintained. Session continuity broke. Retries amplified the failure. Policy and context diverged. The business process was unusable.

The reporting boundary is no longer the operational boundary. The vendor reports the piece. The customer experiences the system. In the modern architecture, the relationship between those two is not transparent. A component can be green while the system is functionally broken. The deception is not in the sentence. The deception is in the boundary chosen for the sentence.

What palliative reporting looks like

The vocabulary itself shows the shift. "Elevated errors." "Degraded performance." "Some customers." "Investigating reports." "Partial impact." "Third-party provider issue." "Service infrastructure not

processing traffic as expected.” That language is calibrated to a sustainment posture, not to a coherence posture. It describes symptoms. It does not describe whether the system held its own integrity through the event.

This is internally consistent with palliative care. The chart records pulse, blood pressure, respiration. It does not record whether the person is still themselves. The framing is appropriate when the underlying condition is sustainment of function, not restoration of health. That is exactly where the industry now operates.

The point is not that vendors are dishonest. Vendors are reporting against the model the industry built. The model presumed components composed into systems and that component health was a reasonable proxy for system health. That presumption was always an approximation. Under the load of agentic AI, retry amplification, cross-vendor coupling, and live policy mutation, the approximation has broken. Component health is no longer a proxy for system health. Reporting it as if it were is what makes the public posture misleading.

What an interaction-status model would have to report

The current public status vocabulary covers component availability, error rates, latency, and partial-impact framing. None of those categories answer the questions a customer actually has during a major incident. An interaction-status model would have to report against a different set of categories.

Did the workflow complete correctly? Not whether the API returned, but whether the operation the customer initiated produced the right downstream state. The April 23 GitHub merge queue defect failed this test while every uptime metric passed.

Did authority remain continuous? Whether identity, policy, entitlement, and admissibility held coherently through the event, or whether different subsystems made different decisions about the same participant during the same window.

Did recovery logic amplify or contain the event? Whether the system's response to the trigger reduced the blast radius or extended it. The Microsoft 365 January 22 outage failed this category explicitly, and Microsoft said so.

Did policy, identity, and context remain coherent? Whether the participant's session retained the rules under which it began, or whether mid-flight mutations broke the assumed governance of the interaction.

Were downstream dependent interactions affected? Whether a coordination event in one provider produced cascading symptoms in others that no single status page would surface. The November 18 Cloudflare event is the canonical case.

Was semantic output degraded even if endpoints returned? Whether the system produced correct output, not merely available output. The Anthropic August and April postmortems are the only public disclosures in any major vendor's history that report against this category, and they do it because the existing reporting model has no slot for it.

These are not abstractions. They are what every customer needs to know during an incident and what no current status page can answer.

The architectural correction

A status page cannot, on its own, produce that report. The reporting boundary follows the operational boundary. To report interaction truth, the system would need to know the boundaries of its interactions. That is not a metric problem. It is an architectural condition.

A live interaction has participants, modalities, features, authorities, and transports that mutate at machine speed. To report whether the interaction survived, the system must hold the interaction as a governed object across those mutations and produce evidence, during execution, of what was admissible and what was delivered. Without that boundary, every incident decomposes into component reports. Every component report passes the truth test in isolation. The interaction is the only thing that can fail as a whole, and the interaction is the only thing the current model cannot represent. That governed-object architecture is what the SSOAR work has been building.

That is the gap.

A vendor cannot report what its architecture cannot govern. The status page is downstream of the architecture. The boundary chosen for the sentence reflects the boundary the system can hold.

When the boundary changes, the sentence changes. Until then, “operational” is a component claim, and the system speaks for itself only at the moments when the components were green and the customer was not.