

What Two Failed Deployments Taught Me About Governing AI Output

Jhon B. Arango · September 18, 2026 · 9–11 minute read

Over two days in September I shipped the same category of defect to production twice. Both times a person caught it before any automated check did. Both times that person was me, looking at my own phone, squinting at a page that an AI and I had built together an hour earlier.

This essay is about what fixed it, because the thing that fixed it was not what I expected. It was not better prompting, more careful review, or a more capable model. It was a boundary — a deterministic check that runs before anything reaches the public, refuses the deploy when it fails, and does not care how confident anyone was.

That distinction is the entire argument of AI governance compressed into something small enough to verify in an afternoon. So it is worth walking through honestly, including the parts that are unflattering.

The Failure

The work was routine: add a dark mode to a marketing site. The site already used CSS custom properties — named colour tokens rather than hard-coded values — which is exactly the structure that makes theming straightforward. Define the palette once, re-point the tokens for night mode, and every component follows.

So that is what was done. Several tokens were re-pointed from dark values to light ones. The change was small, internally consistent, and wrong.

It was wrong because two of those tokens had *two jobs*. They set the colour of text, and they also painted the background of the page header. Re-pointing them for night mode turned the header white while its heading turned dark — and a third element, whose colour was written as a literal rather than a token, stayed white and became invisible against the new background.

The page did not throw an error. Every test that existed still passed. The HTML was valid, the links resolved, the deploy reported success. The only signal that anything was wrong was that a human looked at it.

I fixed that instance, verified it, and shipped again. Two days later a second screenshot arrived showing a different element with the same underlying fault: a text colour sitting on a token that also painted a surface. I had fixed the symptom I had been shown, not the class of defect.

Why More Care Was Not the Answer

The instinctive response to shipping a defect twice is to resolve to be more careful. In my experience this almost never works, and it is worth being precise about why.

Care is a scarce, fluctuating resource. It degrades with fatigue, with familiarity, and — importantly for this discussion — with confidence. The second defect shipped precisely *because* the first had been fixed. Having just reasoned carefully about the token system, I trusted my model of it more than I should have. Confidence rose while actual coverage did not.

This is the same failure mode organisations exhibit when they govern AI through policy alone. A document stating that outputs will be reviewed for accuracy is a statement about intent. It creates no evidence, survives no audit, and degrades silently under deadline pressure. When a regulator asks what the system was permitted to do and what it actually did, a policy cannot answer at that resolution.

The alternative is not to trust less. It is to make trust unnecessary at one specific boundary.

What a Boundary Looks Like

The check I built renders every page of the site at two viewport widths in both colour themes, then examines every element that contains text. For each one it measures the contrast ratio between the text and the surface behind it, and fails if the result falls below the WCAG AA threshold. It also flags anything extending past the viewport, and any page missing its canonical URL, structured data, or accessibility landmarks.

The first run examined roughly 2,250 text elements and reported 147 distinct failures.

That number deserves a moment. I had been staring at this site for two days. I had fixed two reported defects by hand and believed the work was finished. A mechanical check that understands nothing about design, brand, or intent found 147 problems in its first execution — including a navigation button whose label was unreadable against its own background, on every page, in the theme I had personally reviewed.

Crucially, the check does not report. It *refuses*. It is wired into the deployment script ahead of the upload step, and a failure aborts the deploy with a non-zero exit code. I verified this by deliberately introducing a low-contrast rule and confirming the deploy stopped rather than warning and continuing.

That verification matters more than it might appear. A gate nobody has watched fail is not a gate — it is a hope with a log file. Any control you intend to rely on should be tested by breaking it on purpose, at least once, deliberately.

Two Traps Worth Knowing

Building the check surfaced two measurement errors that are worth stating plainly, because both produce *confidently wrong* results rather than obvious failures — and confidently wrong is the dangerous kind.

The first: headless browsers do not necessarily give you the viewport you asked for.

Requesting a 390-pixel-wide window — roughly a modern phone — returned a rendering context about 500 pixels wide, because the browser enforces a minimum window width. Every mobile-specific check silently ran against a desktop-ish layout and passed. Worse, screenshots captured the left 390 pixels of a 500-pixel layout, so elements positioned to the right of that boundary simply were not in the image. I spent a long stretch hunting a control that was rendering perfectly, just outside the frame I was looking at. The fix is to render inside a fixed-width container rather than trusting the window.

The second: you cannot reliably compute a background colour from the stylesheet.

The obvious approach — walk up the element tree until you find a background colour — fails in two directions at once. An element painted with a gradient reports *no* background colour at all, so the walk passes straight through it and returns whatever sits behind. And semi-transparent layers are reported as their own colour rather than the blend a human actually sees. Both errors occurred repeatedly and in both directions: elements flagged as failing when they were fine, and elements passing when they were unreadable.

The reliable method is to stop reasoning about the stylesheet and sample the rendered pixels. Screenshot the page, look at the actual colours inside each element's box, and take the most common one as the background — excluding pixels close to the text colour, or a large headline scores perfectly against its own letterforms.

The general principle generalises well beyond web pages: **measure the artefact, not the intention**. The stylesheet is a statement of intent. The rendered pixels are what the user receives. Where the two disagree, the pixels are the truth, and any check that reads intent will eventually certify something no human would accept.

The Same Fault, Found Twice More

While preparing to publish this piece, I went looking at a second, unrelated publishing pipeline — an agent that drafts long-form articles — and found the identical failure pattern at a different layer.

The agent asks a model to return only an HTML article body. The model usually complies. When it does not, it wraps the article in a code fence, or prefixes it with editorial commentary about the draft it was given. Nothing downstream checked. The response was written into the page template verbatim.

The result was that a substantial majority of published articles rendered with a literal code-fence marker at the top and bottom of the body. A handful were worse: they opened with the model's own critique of the draft, including one instance where the model speculated in print that a methodology referenced in the article might be invented. That text was live, on a professional site, presented as the article.

The fix mirrored the first one exactly. Extract the real content when a fence is present, then *refuse to publish* if scaffolding survives, if the body contains no markup, or if it is implausibly short — validated against every observed failure pattern, including the case where commentary appears with no fence to key off. And, as before, the prompt still politely asks for clean output. It simply is no longer the thing that is relied upon.

The Governance Lesson

Three observations generalise from this, and they are the reason a small accessibility bug is worth an essay.

Ask for compliance; enforce it anyway. Both failures shared one shape: an instruction was issued in a prompt, the model usually followed it, and nothing checked. A prompt is a request. In a system that ships to the public, every request that matters needs a corresponding check that does not depend on the request having been honoured. This is not distrust of the model — it is the same reason a type system exists in a language written by competent people.

The boundary belongs where output becomes irreversible. Not at generation, where the cost of being wrong is a retry; at publication, where the cost is a public artefact and a credibility loss. In both cases the right place was the deployment step — the last moment where refusal is still cheap.

What is not measured is not governed, regardless of what the policy says. I could have written a standard requiring all pages to meet AA contrast. It would have been true, well-intentioned, and worth nothing — the defects shipped in a project where I already knew and believed that standard. The standard became real the moment something mechanical could reject a deploy that violated it.

The uncomfortable implication is that most AI governance programmes are currently at the policy stage and describe themselves as governed. They have principles, a committee, and a review process staffed by attentive people. What they cannot produce, for any particular output, is evidence of what the system was permitted to do, what it actually did, and which check confirmed the difference.

Key Takeaway

AI systems do not fail like traditional software. They fail plausibly. The output is well-formed, confident, and wrong in ways that pass every structural test you already run. That is precisely why governing them through review and intent does not scale: human attention is the one resource that degrades exactly when volume increases.

The practical response is narrower and duller than most governance frameworks suggest. Identify the point where output becomes irreversible. Put one deterministic check there. Make its failure block the action rather than annotate it. Then break it on purpose, once, to confirm it actually blocks.

Everything upstream of that boundary can remain probabilistic, creative, and fast. That is what AI is good at. The boundary is what makes it safe to let it be good at that — and it is the difference between a governance programme that produces documents and one that produces evidence.

Jhon B. Arango · ARKONA Research · www.arkonaresearch.com · jhon.arango@intrepidcyber.ai

Originally published at www.arkonaresearch.com/blog. Reproduction permitted with attribution.