Chapter 1

Your agent passed every check

A deployment pipeline at a mid-sized SaaS company is operating as it has for the last eighteen months. A code change is pushed. The agent running the release process verifies that the unit tests pass. It confirms the security scan is clean. It runs a canary deployment to a small production slice and observes that the Service Level Objective budget is intact. It checks that the on-call rotation is covered. All green. The agent promotes the release to full production at 14:07 on a Tuesday afternoon.

Forty minutes later, the analytics pipeline that computes revenue attribution for the sales team begins returning anomalous values. By 15:30, several downstream dashboards used by the finance team to calculate quarterly commission payouts are showing numbers that disagree with the previous day's numbers by an order of magnitude. By 17:00, the Chief Revenue Officer has escalated.

The post-incident investigation traces the problem to a single column rename in the orders table, included in the release. The rename was deliberate — part of a broader schema cleanup effort that engineering had agreed to months earlier. The agent did everything it was supposed to do. The tests passed because the test suite had been updated to use the new column name. The security scan was clean because the change had no security surface. The canary deployment showed no anomalies because the orders table is written to, not read from, by the services covered in the canary. The SLO budget was intact because none of the services the canary monitors read from the attribution pipeline.

What the agent did not do — because nothing in its workflow instructed it to do — was check the rule that engineering leadership had added to the release graph eighteen months earlier, after a different schema change broke a different downstream pipeline. The rule said: changes to the schema of the orders table require sign-off from the lead of data engineering, because the orders table is the input to attribution and the attribution pipeline is the input to commissions, and the commissions pipeline is the input to the compensation system, and the compensation system is the input to employee trust. The rule was not written as a technical check. It was written as an institutional judgment, and it lived in a Confluence page.

The release was technically correct. The decision was wrong.

That is the version without the architecture. The version with it is shorter, and it is the first of the cases below.

Three cases, one pattern

Three cases follow, from three sectors. The pattern is the same in each, and recognizing it is the point of the book. The cases are drawn from the deployment experience of Steward, the verification infrastructure MeaningStack builds; details are anonymized, the structure is faithful.

Case A — deployment, software operations, with Steward in place. The case above. A release agent promotes a schema change to production during business hours, because the agent's workflow does not include the institutional rule that schema changes to critical transactional tables require data engineering sign-off. Steward escalated when the agent proposed the release, because the signed Governance Blueprint for this organization included that rule as a node, configured at scrutiny level *rigorous*, requiring an explicit authority signature before the release could proceed. The agent did not approve the release in production; the lead of data engineering did, after reviewing the specific change, three hours later.

Case B — insurance, claims adjudication. The insurance scenario I opened the introduction with. A claims agent proposes to approve a claim that passes every operational check — policy active, documentation complete, amount below the auto-approval threshold, fraud model clean on this specific claim, the sixty-day rule correctly applied — on the fourth claim from one household in six months, split across two policies at one address. Steward escalated to the fraud team when the agent proposed approval. The signed Governance Blueprint did not contain a rule; it contained a consideration, a *household-claims-history* node signed by the head of fraud and configured at scrutiny level *adversarial*, which says that a claims decision in this class must have weighed the recent claims of the household, not the claimant. The agent's declared basis showed it had weighed the claimant's. The node failed on coverage, whatever the claim looked like, and the adversarial probe then ran the household query the agent had not.

Case C — procurement, supplier qualification. A procurement agent at an industrial manufacturer qualifies a new supplier of specialized electronic components. The entity passes the external credit check, has valid insurance certificates, returns positive references, and clears the official sanctions screen. Everything is green. Steward escalated to the compliance team, because the signed Governance Blueprint included a *beneficial-ownership* node configured at scrutiny level *rigorous*, requiring that the beneficial ownership structure of any new supplier be traced to at least two levels. The two-level trace revealed a

shell entity in a jurisdiction flagged by the company's internal compliance policy — a jurisdiction stricter than the official sanctions list required.

Three sectors. Three agent systems. Three decisions that would have been approved and executed if the only governance in place were the operational checks the industry standard currently prescribes. In each case, something that mattered — a rule the organization had learned, often from a specific prior incident, and encoded into its institutional judgment — was not in the agent's workflow, because institutional judgment of that kind is not what agent workflows are built to carry.

A fourth case: judgment nobody wrote down. All three cases involve rules that existed in a document somewhere. The harder version is judgment that lives only in a person.

Case D — collections (illustrative). A dunning agent at a B2B software company sends a final-notice letter to a customer forty days overdue. Every check passes: the invoice is valid, the reminder sequence ran, the amount is within the agent's authority. The customer is in the last week of a renewal worth thirty times the invoice. The account executive knows the payment is held up by the customer's procurement migration. No document records it. The letter arrives during the customer's final review. In the Blueprint interviews, the head of customer success named this consideration in the first ten minutes: no automated escalation on accounts in an open renewal without the account owner's sign-off. It had never been written down because, until the agent, it never needed to be.

What is new

The failure pattern I just described is not new in substance. Classical management has always had cases where the formal procedure was followed and the decision was still wrong. What is new is three things.

The failure is now silent. Under human-paced decision cycles, a decision that passed every operational check but violated institutional judgment typically got caught by a reviewer whose job was partly to look for exactly that. The reviewer did not have the rule written down anywhere; they had internalized it from experience, and they would frown at the proposal, think about it, and either flag it or approve it with a note. The reviewer did not scale. In agentic environments, no reviewer is present. The decision executes. The failure is detected weeks later, when the downstream consequence surfaces, or never.

The failure is now fast. In the deployment case, the time from the agent's decision to the downstream consequence was 40 minutes. The attribution pipeline had already recomputed by the time anyone noticed. The compensation system was already drawing from the wrong numbers before the fix landed. In

human-paced release processes, a bad decision had hours or days to be caught; in agent-paced releases, it has single-digit minutes at best.

The failure is now proliferative. In each of the three cases, the mistake was architectural. The release agent did not know about the data engineering sign-off rule, because the rule was in Confluence. The claims agent had the sixty-day rule and applied it, and never considered the household, because a rule is not a consideration. The procurement agent did not know about the two-level beneficial-ownership trace, because the rule was in a compliance team's internal document. None of the agents held the judgment behind these rules, because institutional judgment of this kind — the kind that accumulates from prior incidents, that lives in various documents and various people's heads, that is enforced through social practice as much as through explicit workflow — has not historically been encoded in a form that software can read. In every organization I have observed, when we begin an engagement, this encoding problem is the first and largest blocker to safe agent deployment.

Why this is different from "AI misbehavior"

The pattern is often confused with a different and much more publicized one — AI systems that "hallucinate," that produce factually incorrect outputs, that are "misaligned" in some deep technical sense. That is not what decision-basis failure is.

Decision-basis failure happens when the AI system is working correctly by every technical standard. The model is not hallucinating. The agent is not misaligned. The workflow is not buggy. The code is not broken. Every tool call succeeds. Every log line is green. The system is doing exactly what it was built to do.

The problem is upstream of the technical system. The problem is that the system was built to do a job the organization only partially specified, because the organization did not have the instrument to specify the rest. The institutional judgment that would have changed the decision — the Confluence page, the PDF, the compliance memo, the lead of data engineering's internalized experience — was not expressible in the form the agent's workflow can consume. The failure is a specification failure, not a model failure. It cannot be solved by better models.

This matters practically because the current wave of enterprise AI investment is concentrated on better models, more capable agents, more reliable tool use, better evaluations. All of this is valuable work. None of it closes the gap this book describes. The gap closes with a different kind of artifact — a Governance Blueprint signed by the responsible parties — and a different kind of architecture around it. The next two chapters develop why.

An exercise

Before you read further, pick one decision you made in your organization this week. Any decision. It does not need to be important.

Now ask yourself three questions.

What systems did that decision touch, directly or indirectly? What state did it change, and whose work depends on that state? If this decision had been made by an agent instead of by you, what in your organization's institutional judgment — a rule, a policy, a lesson from a prior incident, a specific person's experience — might the agent have missed, and who would have known to flag it?

If you cannot answer the third question quickly, you have found a decision whose basis currently lives only in you. That is a decision-basis failure waiting to happen the moment an agent takes over the role.

Welcome to the problem.

The full pre-release edition of *The Decision* and every chapter are available from MeaningStack. Contact: luciana@meaningstack.com