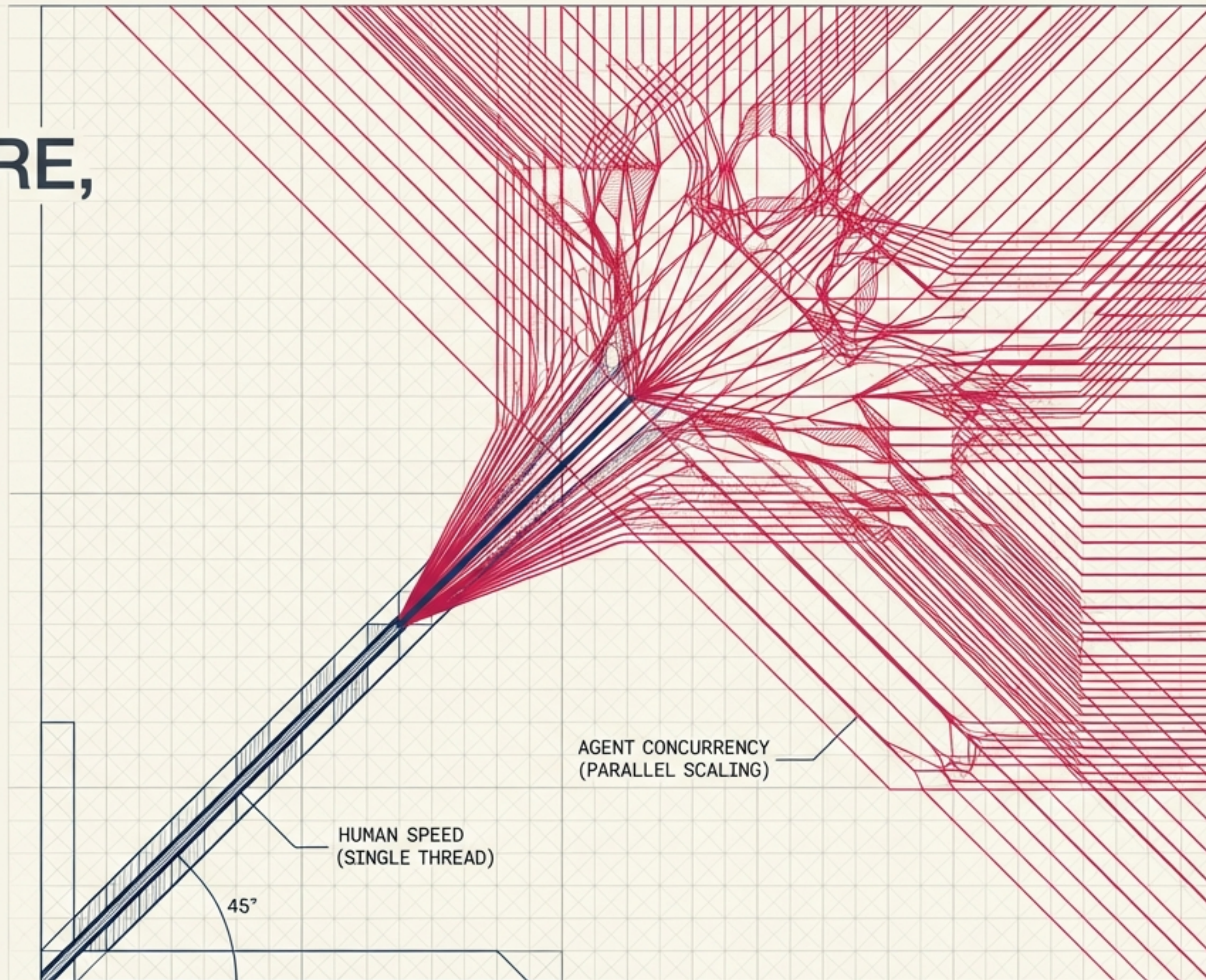


HUMAN-ERA INFRASTRUCTURE, AGENT-ERA WORKFLOWS

A Diagnostic Report:
Rebuilding the software stack
for autonomous systems.

Based on the architectural field
notes of Thor Henning Hetland.



Human-Era (Git)

10:00 AM

2:30 PM

4:45 PM

Agent-Era (Jujutsu / JJHub)



```
10:00:01.001  
10:00:01.002  
10:00:01.003  
10:00:01.004  
10:00:01.005  
10:00:01.006  
10:00:01.007  
10:00:01.008  
10:00:01.009  
10:00:01.010  
10:00:01.011  
10:00:01.012  
10:00:01.013  
10:00:01.016  
10:00:01.017  
10:00:01.018  
10:00:01.019  
10:00:01.020  
10:00:01.021  
10:00:01.022  
10:00:01.023  
10:00:01.024  
10:00:01.025  
10:00:01.026  
10:00:01.027  
10:00:01.028  
10:00:01.029  
10:00:01.030  
10:00:01.031  
10:00:01.032  
10:00:01.033  
10:00:01.034  
10:00:01.035  
10:00:01.036  
10:00:01.037  
10:00:01.038  
10:00:01.039  
10:00:01.040  
10:00:01.041  
10:00:01.042  
10:00:01.043  
10:00:01.044  
10:00:01.045  
10:00:01.046  
10:00:01.047  
10:00:01.048  
10:00:01.049  
10:00:01.050  
10:00:01.051  
10:00:01.052  
10:00:01.053  
10:00:01.054  
10:00:01.055  
10:00:01.056  
10:00:01.057  
10:00:01.058  
10:00:01.059  
10:00:01.060  
10:00:01.061  
10:00:01.062  
10:00:01.063  
10:00:01.064  
10:00:01.065  
10:00:01.066  
10:00:01.067  
10:00:01.068  
10:00:01.069  
10:00:01.070  
10:00:01.071  
10:00:01.072  
10:00:01.073  
10:00:01.074  
10:00:01.075  
10:00:01.076  
10:00:01.077  
10:00:01.078  
10:00:01.079  
10:00:01.080  
10:00:01.081  
10:00:01.082  
10:00:01.083  
10:00:01.084  
10:00:01.085  
10:00:01.086  
10:00:01.087  
10:00:01.088  
10:00:01.089  
10:00:01.090  
10:00:01.091  
10:00:01.092  
10:00:01.093  
10:00:01.094  
10:00:01.095  
10:00:01.096  
10:00:01.097  
10:00:01.098  
10:00:01.099  
10:00:01.100
```

THE SYMPTOM

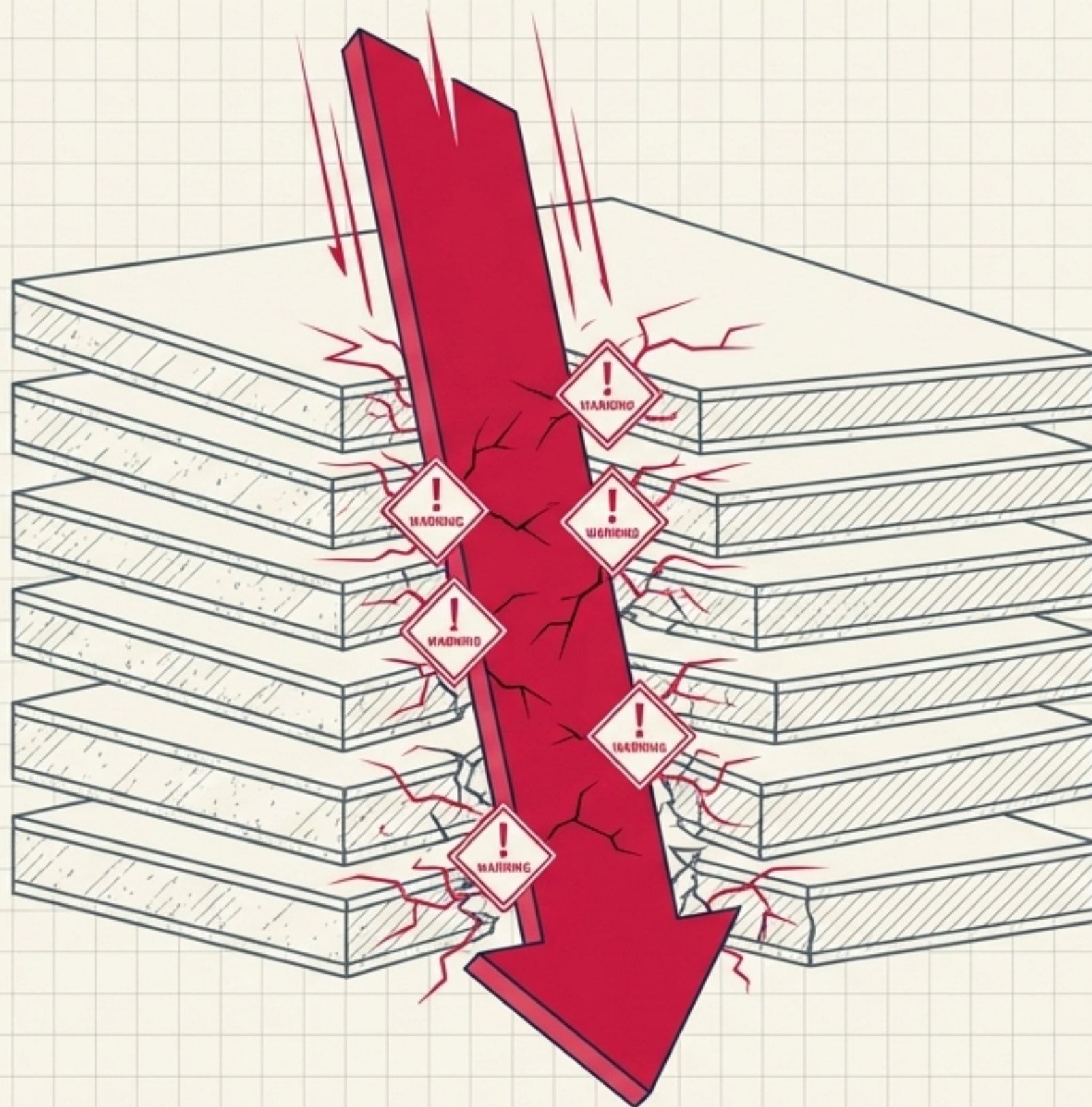
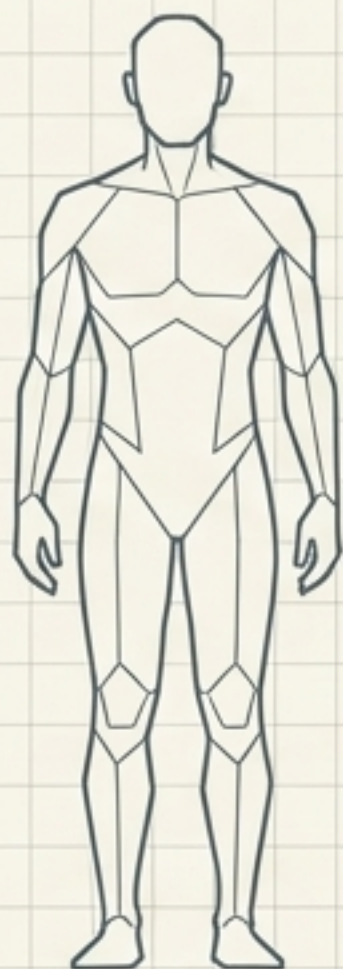
JJHub recently launched a version control platform built on Jujutsu instead of Git, specifically for agentic engineering teams.

THE CAUSE

Traditional Git expects a human commit cadence. Agents produce concurrent, high-frequency operations that shatter those expectations.

THE IMPLICATION

This observation is narrowly about version control, but the pattern it reveals is systemic.



Agents are exposing every layer of the stack as a bottleneck.

The entire software development stack was built for humans. Every layer is heavily optimized for:

- Human processing speeds
- Human context windows
- Human coordination patterns

LAYER 5: Integration & Middleware

LAYER 4: Application Logic & Services

LAYER 3: Networking & Security

LAYER 2: Data & Storage

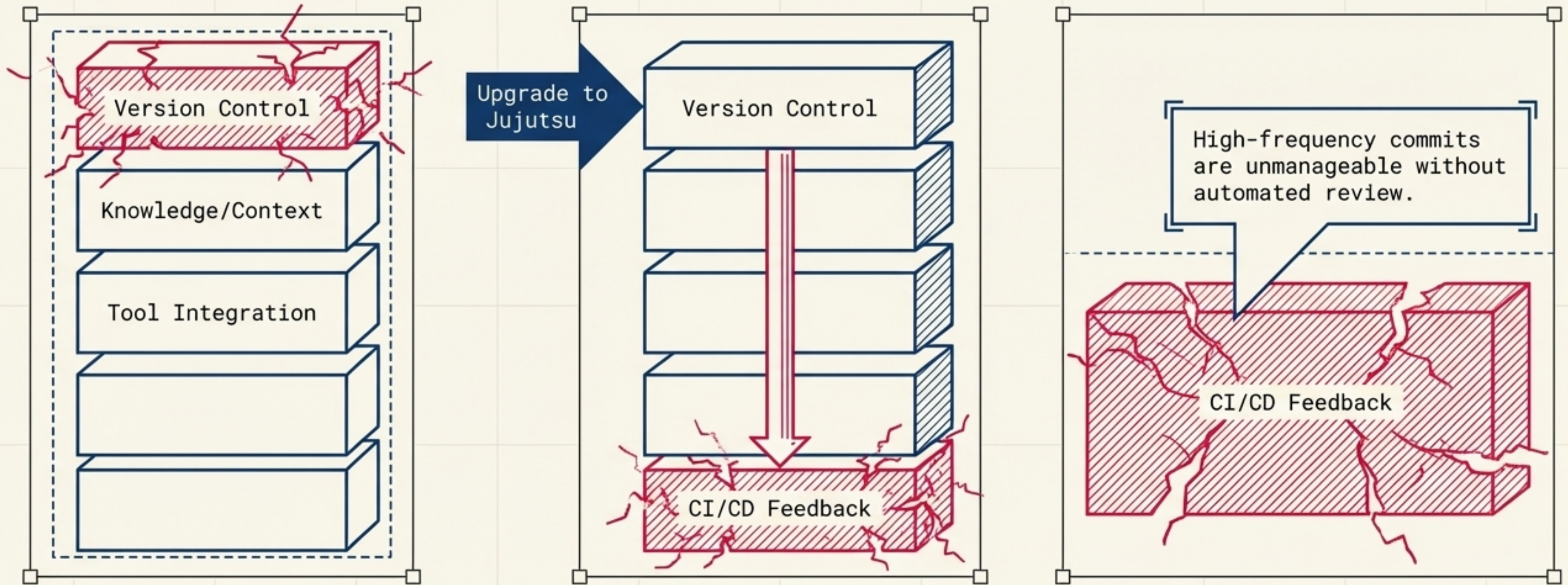
LAYER 1: Infrastructure & Compute

Most engineering teams do not realize they are running agent-era workflows on human-era infrastructure.

The Infrastructure Paradigm Matrix

Layer	Human-Era Assumption	Agent-Era Requirement
Version Control	Git (human commit cadence)	Jujutsu / JJHub (high-frequency, concurrent)
Knowledge / Context	READMEs, wikis, onboarding docs	Structured Context Protocols (KCP, etc.)
Tool Integration	SDKs, manual documentation	Model Context Protocol (MCP)
Orchestration	Manual workflow coordination	CrewAI, autogen, smolagents
Identity / Auth	OAuth, personal access tokens	SPIFFE, AgentControl
CI/CD Feedback	Human code review	Agentic CI (automated structural judgment)

The Bottleneck Cascade: Upgrading one layer moves the constraint to the next.

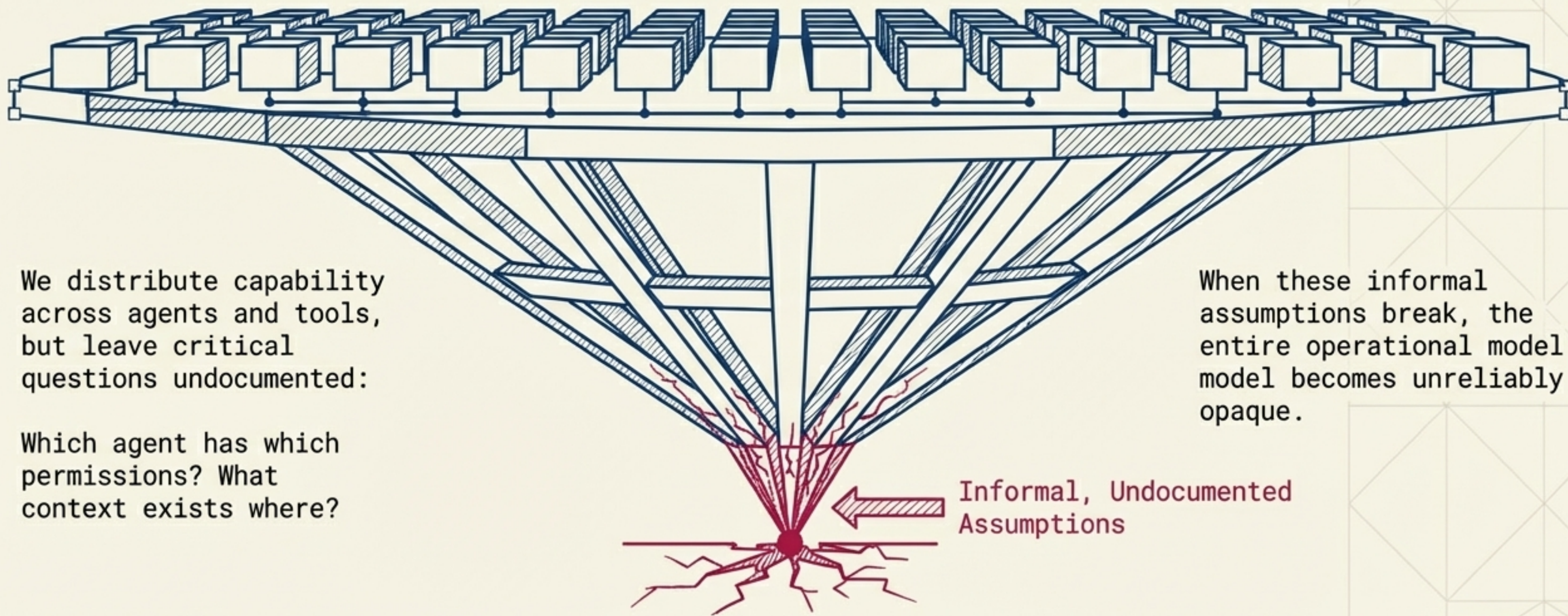


No single layer is the bottleneck. Because they were all designed together for human-speed operation, the layers are co-dependent. Treating the agentic stack as six separate tool choices guarantees cascading failure.

The Wiggins Fragility Paradox: Consolidated Fragility

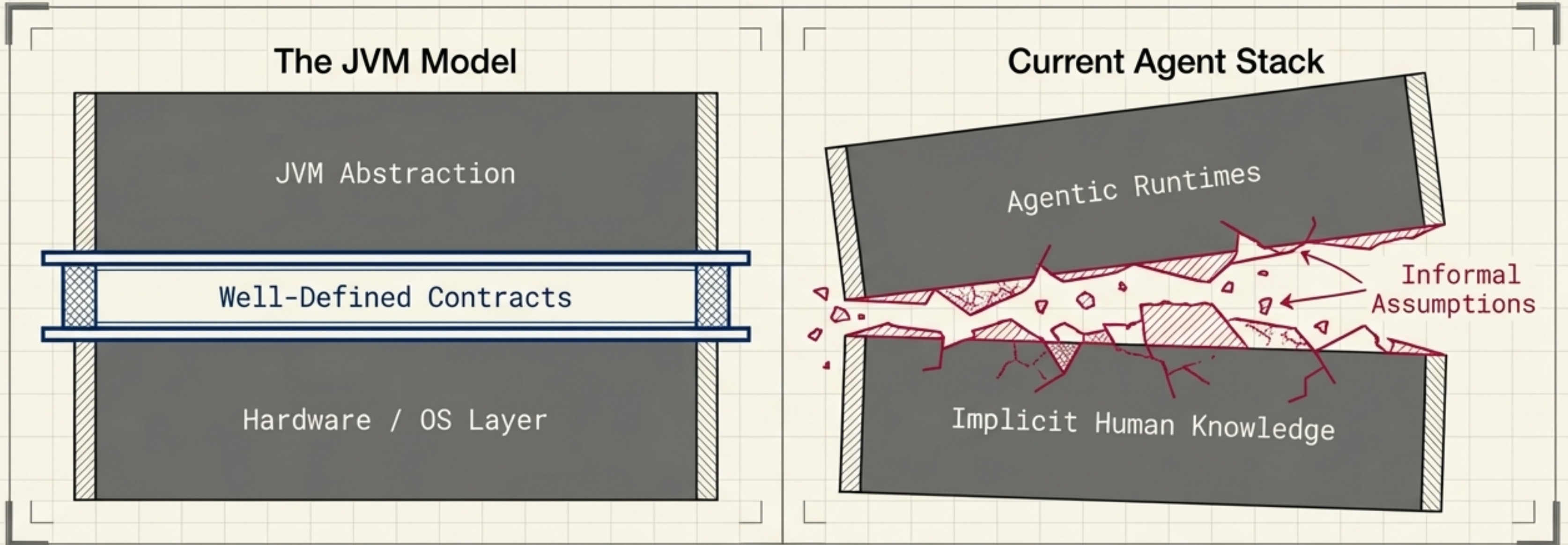
Systems that appear distributed but concentrate operational capacity at a few shared assumptions. — Dion Wiggins

Highly Distributed Capability: Agents, MCP Protocols, Orchestration Frameworks, Tools



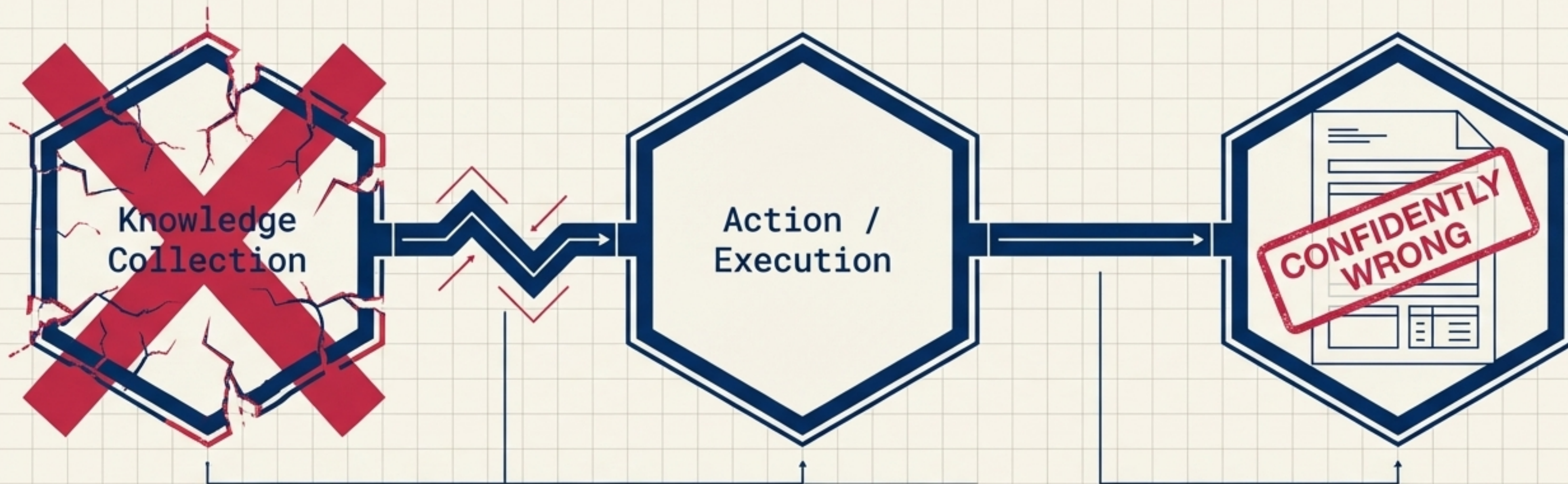
Agent systems run on undocumented runtimes, not virtual machines

- Steve Jones points out that abstractions like the JVM only work because the layer below has rigid, well-defined contracts.



- By contrast, current agentic runtimes rely on informal, undocumented behavior. It is impossible to build reliable abstractions on top of implicit human knowledge.

The Lindenberg Rule of Ingestion: Failures happen upstream of the action.



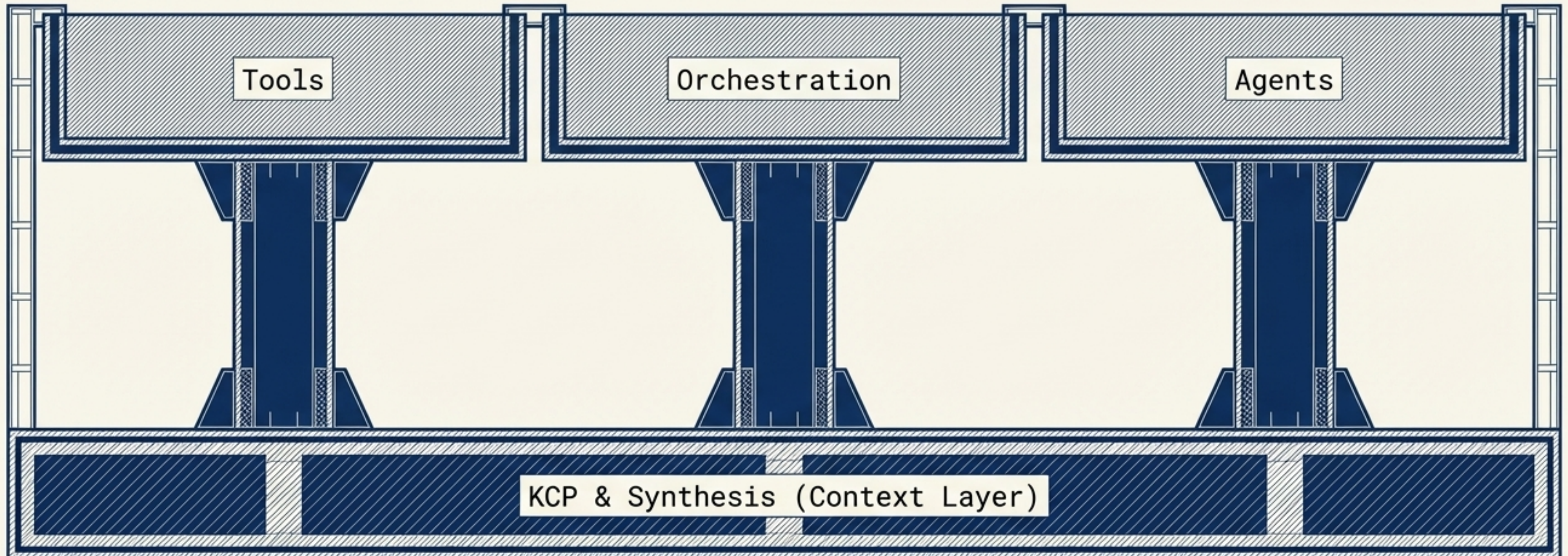
Agents rarely fail at the point of action. An agent with access to powerful tools but insufficient context will simply make confidently wrong decisions at lightning speed.

The failure is ingestion. Structured, machine-readable context is not a nice-to-have. It is load-bearing infrastructure.

Context protocols are load-bearing infrastructure.

Building at the knowledge/context layer reveals profound dependencies. If the context isn't structured and machine-readable, every other layer in the stack operates blindly.

Open specifications like KCP and utilities like Synthesis are designed to replace informal wikis with explicit, structural data flows.



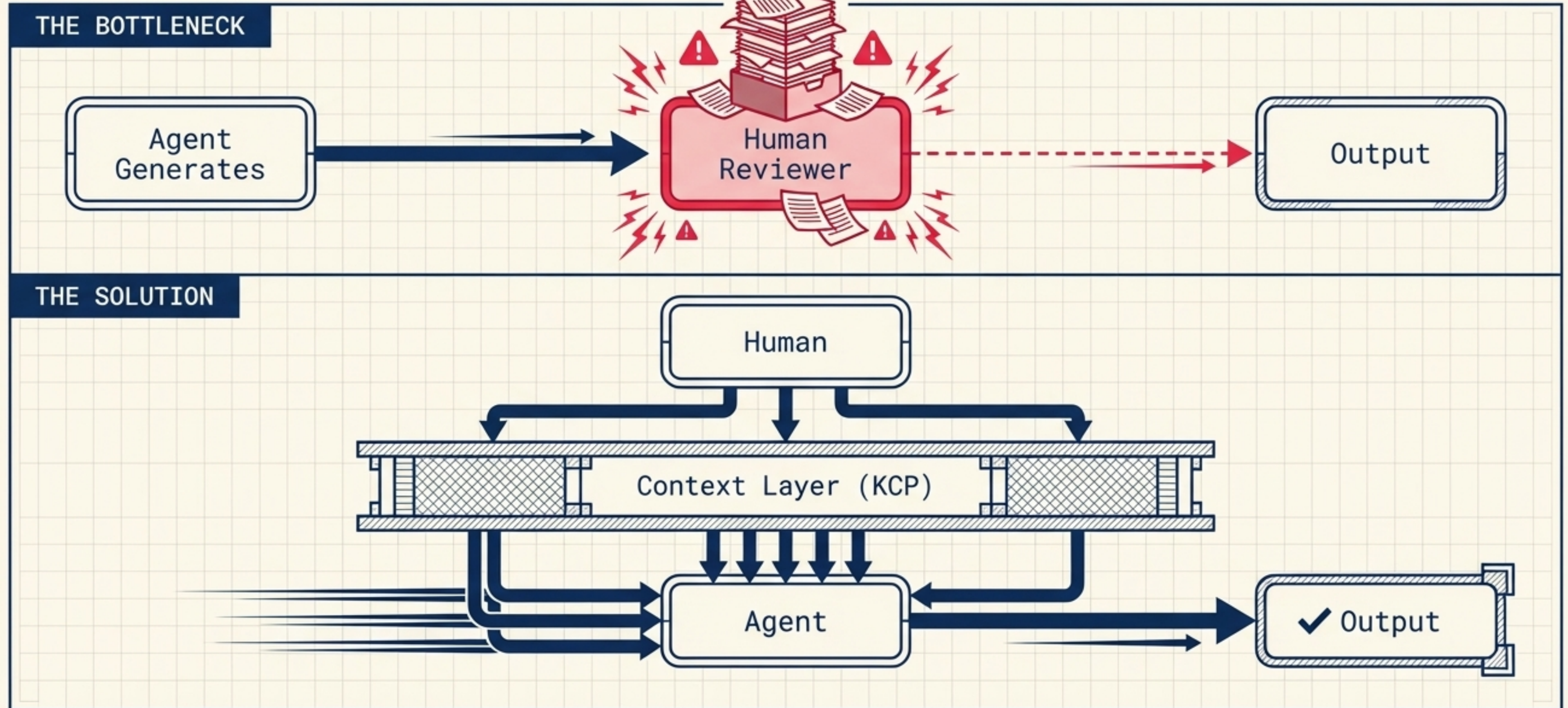
The Andresen Imperative: Encode judgment before the loop starts.

Procedural Guardrails (Human-in-the-Loop)		Structural Guardrails (Encoded Judgment)	
Preserves judgment as a human advantage.	✗	Encodes judgment as an architectural principle.	✓
Human acts as a chokepoint / rubber stamp.	✗	Rules dictate quality, compliance, and risk thresholds.	✓
Breaks under agent speed and commit volume.	✗	Scales infinitely via the context layer.	✓
Tacit knowledge remains hidden.	✗	Forces tacit knowledge to become explicit.	✓

Judgment, aesthetics, and deliberative reasoning need to be encoded before agents act, not preserved as human competitive advantages. – Lasse Andresen

Moving from procedural bottlenecks to structural context.

If important decisions happen at agent volume, the human-in-the-loop becomes a bottleneck. The alternative is the hard work of making tacit knowledge explicit within the context layer.



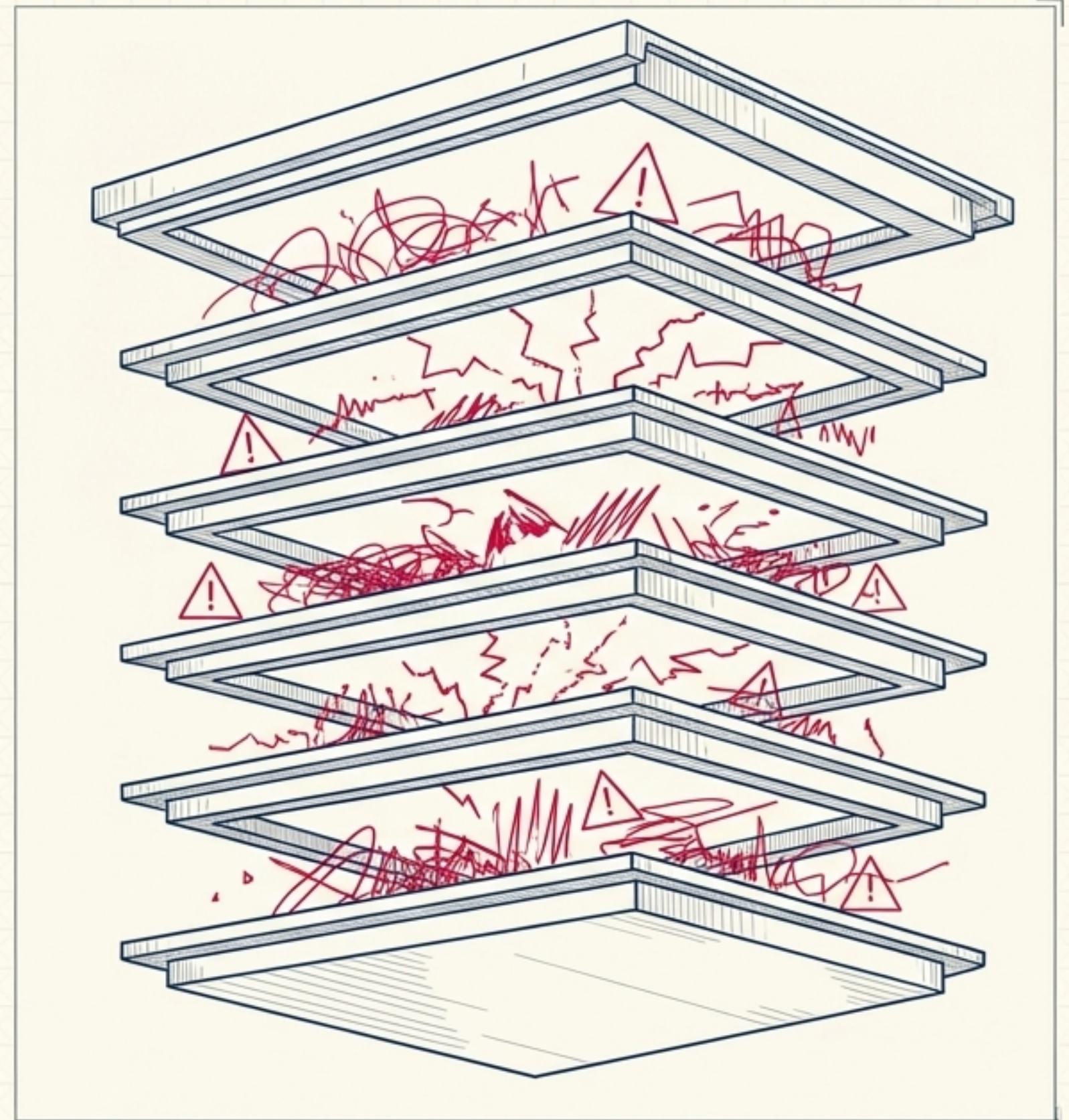
The Real Constraint: Who is working on the seams?

We are in a window where every layer is being rebuilt simultaneously, but in isolation.

- Version control teams are solving version control.
- Context protocol teams are solving context.
- Auth teams are solving auth.

⚠ The critical failure points are the integration gaps between layers. The undocumented contracts. The informal assumptions.

Technical Crimson Roboto Mono



The Seams Diagram

The Coherent Stack Advantage

The layers of the agentic infrastructure stack are not independent tool choices. They are a single, co-dependent problem.

The engineering teams that figure this out earliest—treating the stack as a cohesive integration challenge rather than isolated upgrades—will compound their advantage in ways that are impossible to catch up to.

