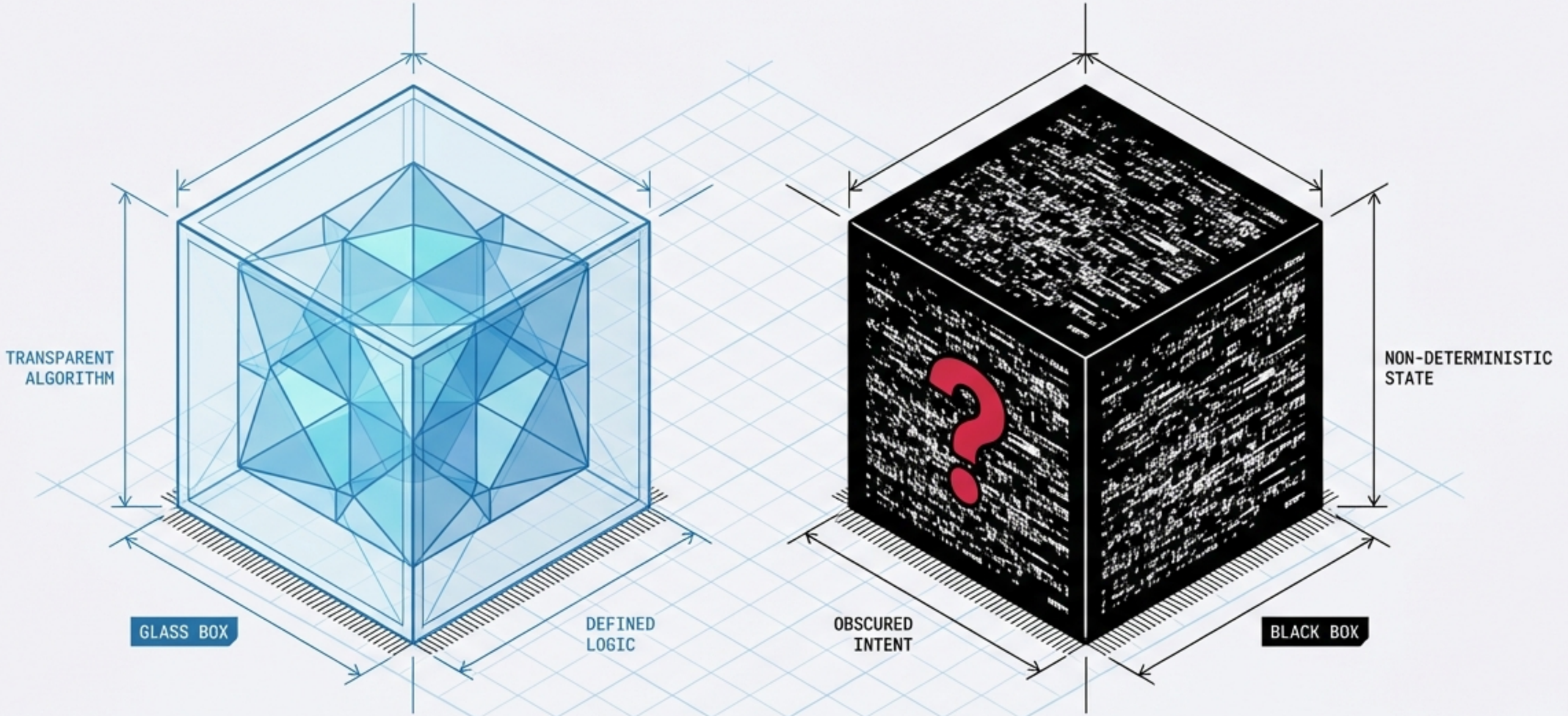


The Unknowable Code

Mitigating Software Liability in the Age of AI

TECHNICAL REPORT: 2026-01-31
AGENTBEATS SUBMISSION
STATUS: VALIDATED

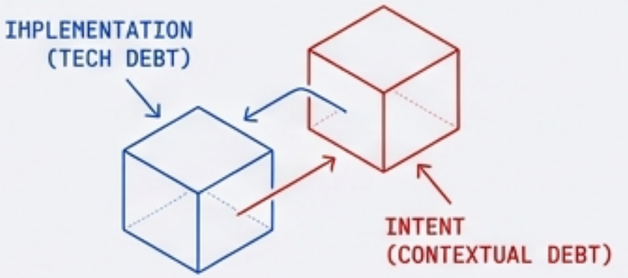
Executive Summary

THE EMERGING CRISIS OF INTENT

01

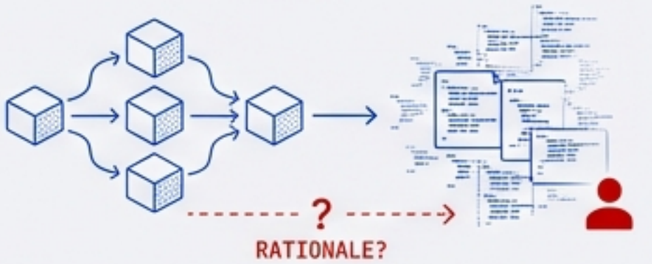
The Shift

For 20 years, the industry managed **Technical Debt** (a failure of implementation). The next decade's liability is **Contextual Debt**—a failure of intent. We are building "Amnesiac Systems" that work, but have forgotten why they exist.



The Catalyst

Modern stacks—AI co-pilots, microservices, and rapid scaling—are eroding human rationale. Code is generated faster than the "theory" of its operation can be built.



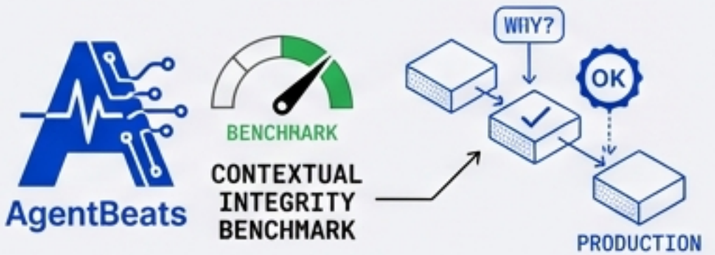
The Stakes

This is no longer an engineering inefficiency. It is a board-level liability, a security threat, and a legal risk under new "Duty of Care" regulations.

BOARD LIABILITY | SECURITY RISK | REGULATORY COMPLIANCE

The Solution

AgentBeats introduces the **Contextual Integrity Benchmark**—the first empirical method to measure and govern the "why" of software before it enters production.



Two Distinct Classes of Software Liability

	Technical Debt	Contextual Debt
Nature of Failure	Failure of the **HOW** (Implementation)	Failure of the **WHY** (Intent)
Core Metaphor	Financial Loan (Speed vs. Quality)	Organizational Amnesia (Loss of Purpose)
Symptom	Spaghetti code. Slow to **change**.	Unknowable code. Dangerous to **touch**.
Remedy	Refactoring & Cleanup	Archeological Code Digs / AgentBeats

"Technical debt makes a system difficult to change. Contextual debt makes it dangerous to touch."

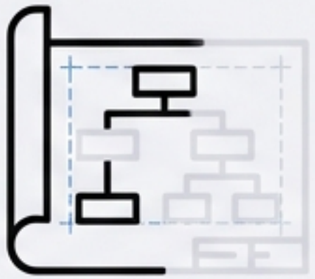
The Anatomy of “Amnesiac Systems”

Liability accumulates on three pillars of missing context.

Missing Intent

The Question: “Why did we build this rule?”

Risk: Modifying algorithms without knowing if they support compliance or temporary marketing. Loss of business goals.



Missing Rationale

The Question: “Why did we choose this architecture?”

Risk: Documented tradeoffs are lost. Architecture becomes a “sacred” artifact. Teams fear violating hidden constraints.

Missing Domain Knowledge

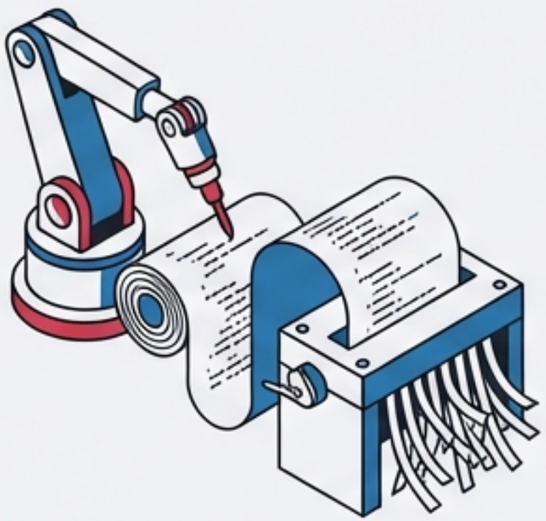
The Question: “What is the difference between a Customer and a Prospect?”

Risk: Business nuance evaporates. Logic errors occur that valid syntax checks cannot find.

The Accelerants

Why the crisis is exploding now.

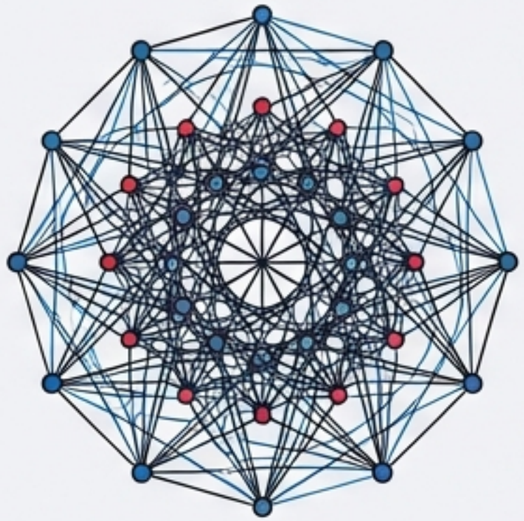
1



Comprehension Debt

Code is generated without a mental model. The 'theory' is lost at the moment of creation.

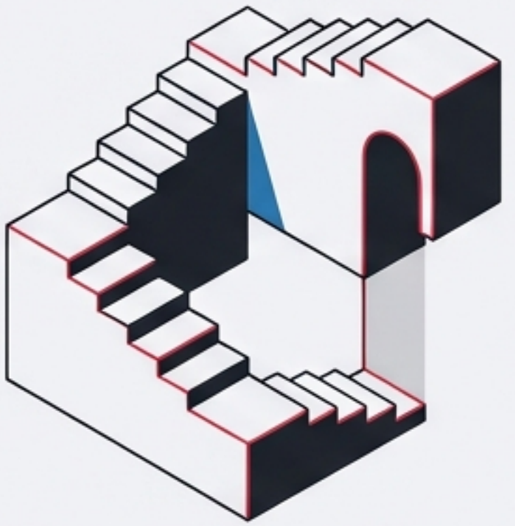
2



The Distributed Monolith

Context is shattered across siloed services. De jure separate, de facto coupled.

3



Premature Complexity

Labyrinthine architectures for hypothetical scale become unknowable liabilities.

RESULT: ACCELERATED ACCUMULATION OF UNKNOWABLE CODE

The ‘Interest Payments’ on Contextual Debt

Operational drag is measurable and severe.

PRODUCTIVITY LOSS

-19%

Compared to teams with high documentation quality.

DEFECT RATE INCREASE

+21%

Unknowable code breeds regression bugs.

ONBOARDING LAG

3 WEEKS

Additional time to absorb tribal knowledge.

REPLACEMENT COST

$50k - $100k

Cost per senior developer lost to burnout.

SLOWER TIME-TO-MARKET

18%

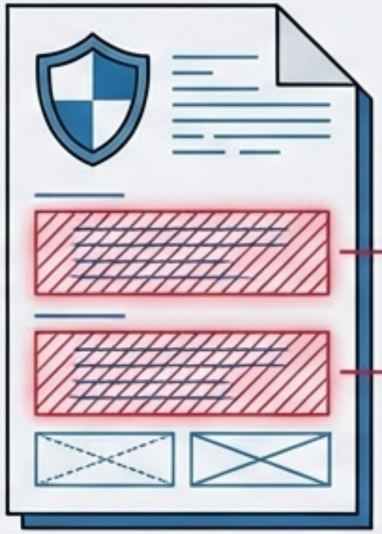
Delay in releasing new features.

The Security Threat: Unknowable is Unsecurable

The Injection of Liability

AI Agent

Hallucinated Rules



Security Policy / FGA Model

Perfectly Disguised Vulnerabilities.

Logic flaws that look like valid code but violate business intent.

The FGA Paradox

Fine-Grained Authorization requires deep domain knowledge.
AI has none.

Result:
Authorization rules that are syntactically correct but functionally insecure.

You cannot secure what you do not understand.

The End of the Liability Shield

Global policy is moving from Strict Liability to Negligence.

The Regulatory Shift

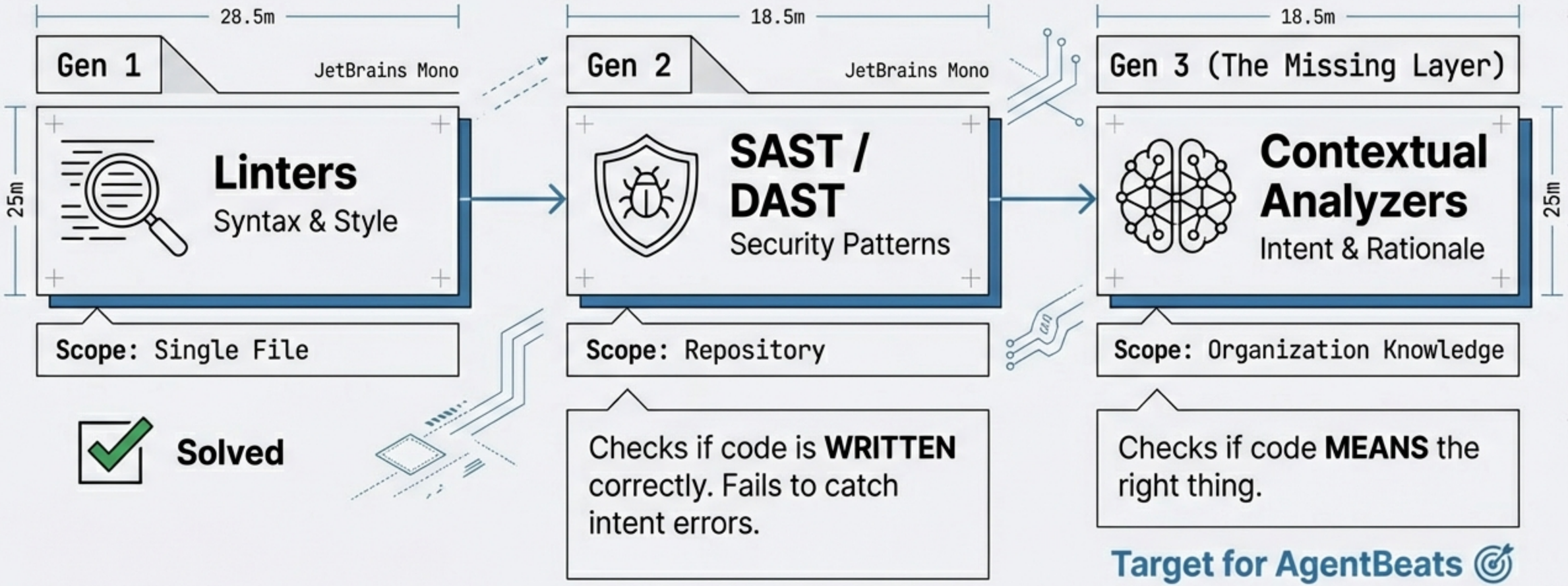
US National Cybersecurity Strategy (2023): "Vendors liable when failing '**Duty of Care**'."

EU Cyber Resilience Act: "Liability for products with digital elements."

High Contextual Debt is **prima facie evidence of negligence**

If you cannot explain **WHY** your system behaves the way it does, you cannot claim you exercised "reasonable precautions". In a breach, you **must produce evidence of intent**, not just logs of activity.

The Tooling Gap: Why Current Tech Fails



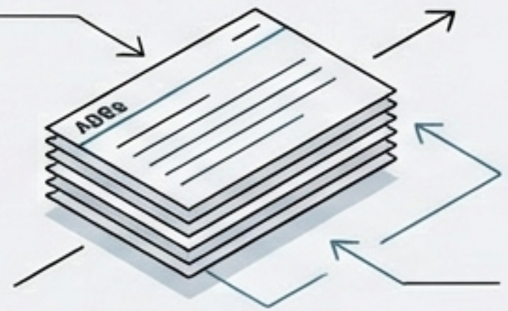
The Discipline: Building Systems with Memory

Strategies to codify the “Why” before automation.

28.5m

ADRs

JetBrains Mono



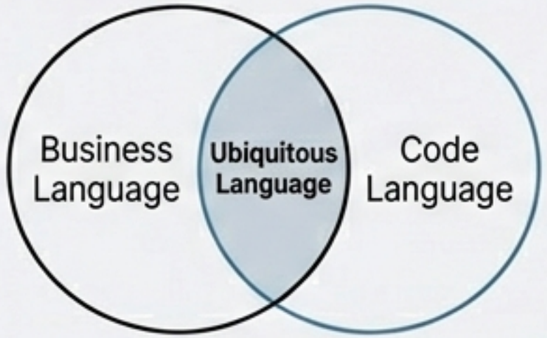
25m

Immutable logs of decisions. Captures **Context**, **Decision**, and **Consequences**.

18.5m

DDD

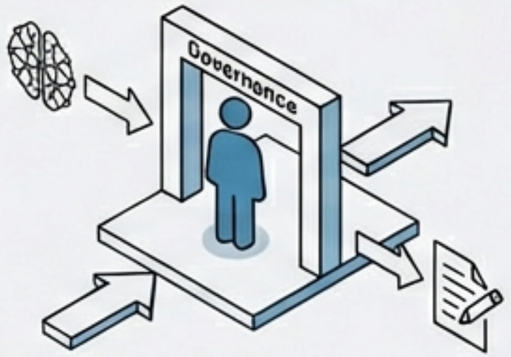
JetBrains Mono



Ubiquitous Language. Code vocabulary must match business vocabulary (e.g., “PremiumCustomer” class).

18.5m

Human-in-the-Loop Governance



25m

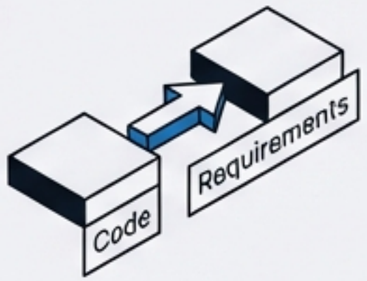
Developers shift from “**Generators**” to “**Editors**”. **Governance** of AI output.

The Contextual Integrity Score (CIS)

Turning abstract liability into a measurable metric.

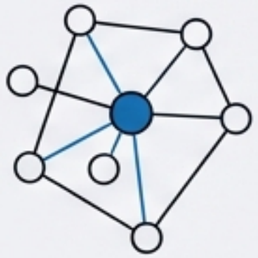
$$CIS(\Delta) = 0.25R + 0.25A + 0.25T + 0.25L$$

R: Rationale



Vector similarity between code and requirements. Includes "Semantic Drift" detection.

A: Architectural



Graph centrality. Does the change violate dependency constraints?

T: Testing

Verification (Pass rate + Specificity bonus)

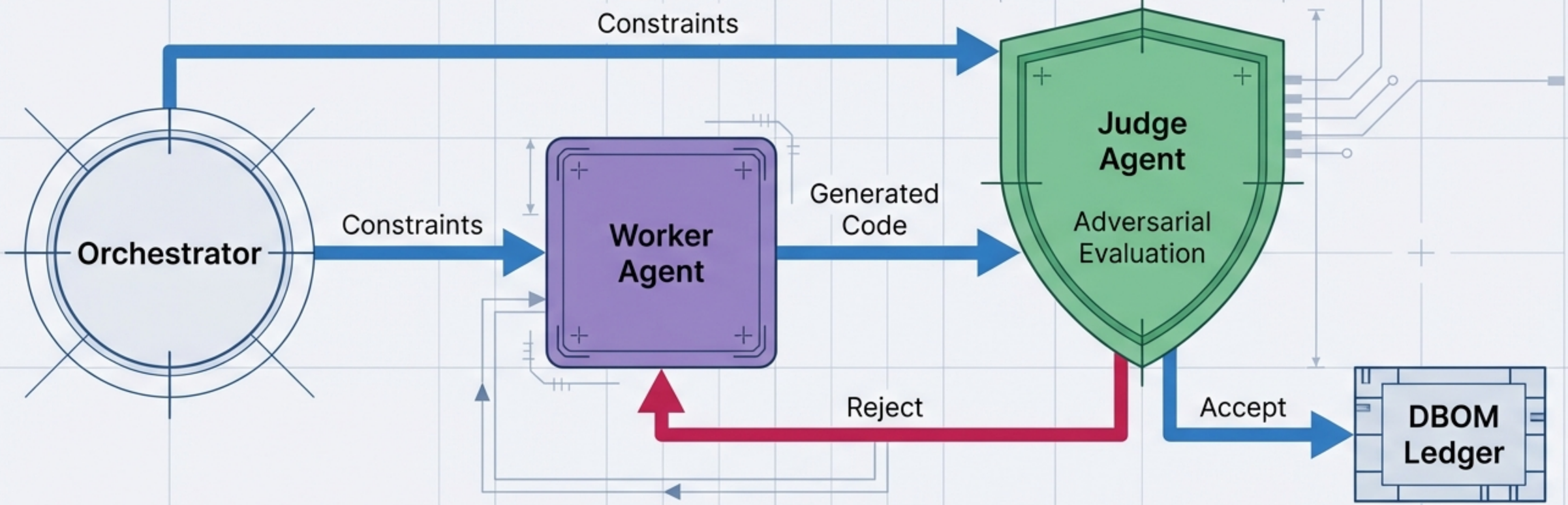
L: Logic

Senior Review LLM. Checks edge cases and complexity.

Goal: A "Fair Isaac" credit score for software integrity.

The Architecture: Agent-as-a-Judge

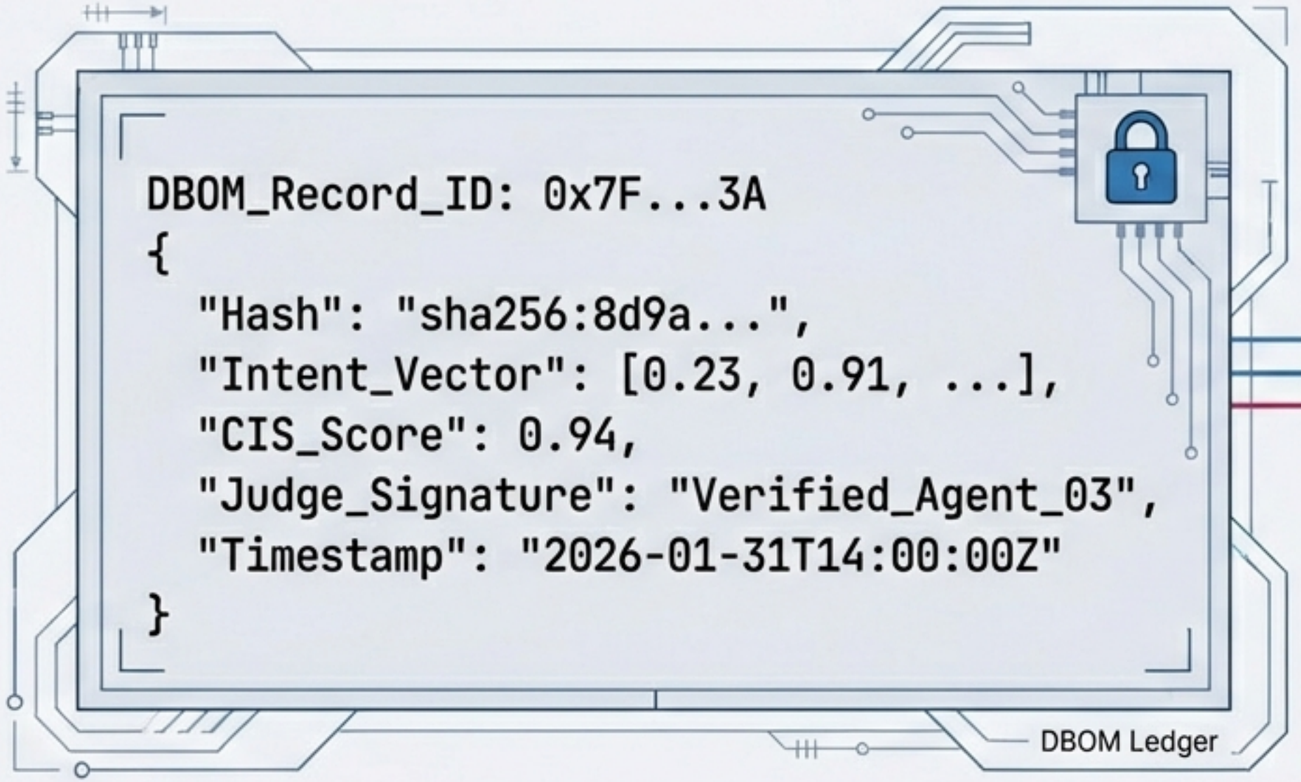
The “Glass Box” Protocol.



Key Concept: Governance Velocity (V_{gov}) must equal Generation Velocity (V_g).

The Artifact: Decision Bill of Materials (DBOM)

From “What is in the code” (SBOM) to “Why it is there” (DBOM).



Immutable: Append-only log.

Traceable: Reasoning trace for every authorized change.

Legal Value: In a lawsuit, this is cryptographic proof of diligence.

The Business Model: The Consult-to-Product Flywheel

A self-reinforcing cycle of high-fidelity services, data acquisition, and scalable SaaS platforms.

Consulting

`Archeological Code Digs.`

High-margin services to reverse-engineer intent for enterprise clients.

Product

`Scalable SaaS Platform.`

Automated Contextual Debt Analyzer (AgentBeats).

Data

`Proprietary Dataset.`

Building the world's only dataset dataset of "Intent Failure" and "Contextual Debt".

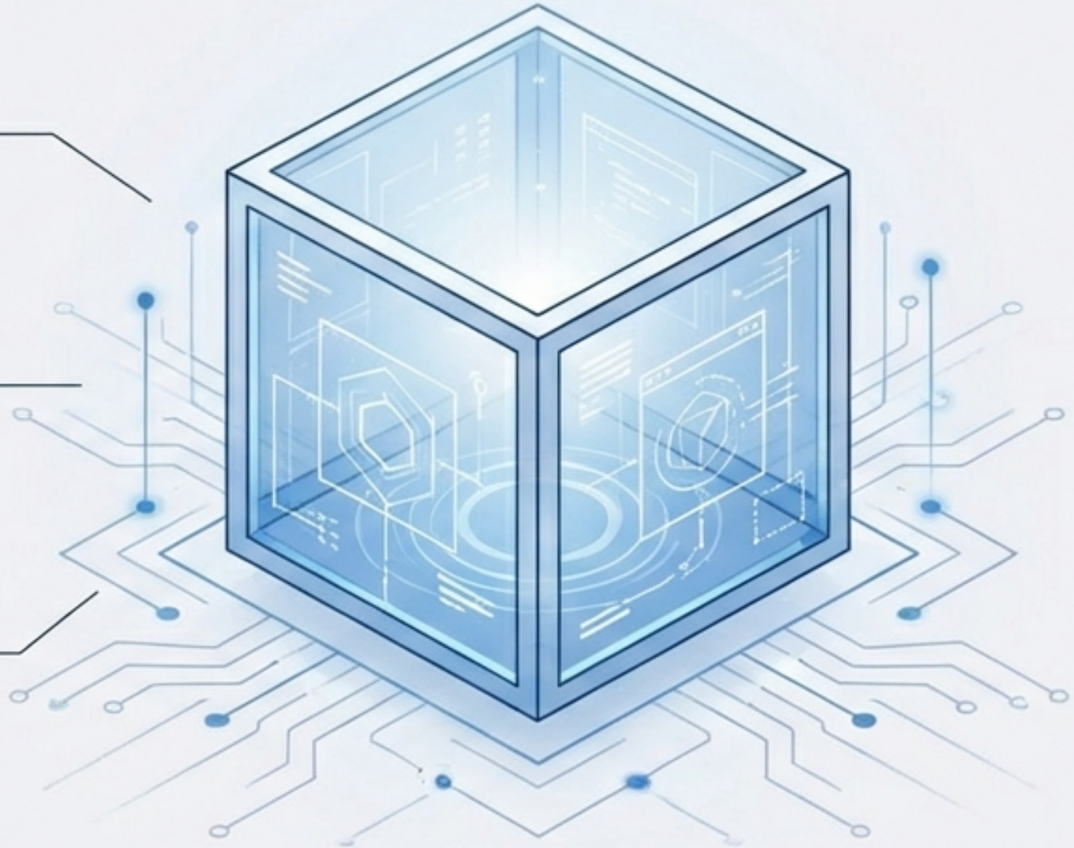
⚠ AgentBeats is not just a developer tool; it is Enterprise Risk Management.

Navigating the Decade of Unknowable Code

The Shift: Contextual Debt is the new Technical Debt.

The Risk: Liability is shifting to the vendor (Duty of Care).

The Solution: Automated "Glass Box" governance to preserve intent.



AgentBeats: The Standard for Contextual Integrity.