A Layered Taxonomy of Agentic AI Failure Modes: Orthogonal Patterns and Three Corrective Primitives for Enterprise Governance

Rajesh Iyer
iyer70@gmail.com

Abstract

As agentic AI systems transition from research prototypes to enterprise deployment, the absence of comprehensive failure mode frameworks impedes systematic risk assessment. This paper presents a practitioner taxonomy of failure patterns organized into a four-layer architectural hierarchy: Cognitive Core, Execution Interface, Interaction Boundary, and Adversarial Surface. We establish orthogonality criteria to minimize conceptual overlap, enabling structured risk quantification. Beyond enumeration, we propose three corrective primitives—Causal State Fabric, Verified Bounded Execution, and Continuous Teleological Audit—as architectural interventions addressing the majority of catalogued failures. The Causal State Fabric, extending Semantic MVCC with provenance, causality, and uncertainty tracking, serves as the foundation upon which the other primitives depend. We argue that traditional quarterly governance cycles are mismatched to agentic failure dynamics, motivating continuous monitoring infrastructure. The taxonomy maps to regulatory frameworks including SR 11-7 model risk management and NAIC insurance AI guidance. This is a practitioner framework, not an empirically validated research contribution; we discuss limitations and validation requirements throughout.

1 Introduction

1.1 Why Enterprise AI Is a Different Game

The dominant discourse on agentic AI failure modes emerges from technology companies. Microsoft, Google, and the AI labs produce taxonomies optimized for their context: clean data models, digital-native entities, and debugging-centric success metrics. This work is valuable but incomplete.

The problem: technology companies don't appreciate the mess. Enterprise data is ugly in ways that don't register in Silicon Valley. And enterprise technology providers—the consulting firms, systems integrators, and software vendors who should understand this mess—aren't picking up the baton. They're waiting for the AI labs to solve it. The AI labs never will, because they don't see it.

Three examples of what gets missed:

The Entity Disambiguation Problem. A coding assistant operates over well-defined entities: files, functions, repositories. An insurance claims agent confronts a different reality: the same policyholder exists as seventeen records across four systems, with name variations, address histories, and no canonical identifier. The same "vehicle" is a VIN in policy admin, a license plate in claims, a telematics ID in IoT, and a handwritten description in an accident report. Agentic systems assuming entity resolution is solved inherit failure modes invisible to technology-sector taxonomies.

The Non-Digital Substrate. Enterprise data is not born digital. Insurance runs on paper forms, faxed medical records, recorded calls, handwritten notes, photographs. Banking processes wet signatures and notarized documents existing as scanned images with OCR artifacts. When an agent reasons over this substrate, failures emerge from the gap between linguistic representation and physical reality—a gap that doesn't exist when operating over code or APIs.

Compliance vs. Debugging. Technology asks: "Does the agent work?" Regulated enterprise asks: "Can we prove to the examiner that it worked correctly, for the right reasons, without discriminatory impact, with appropriate oversight, and with audit trails sufficient for reconstruction three years hence?" SR 11-7, NAIC, Solvency II don't care about 95% task completion. They care about *demonstrated* detection, bounding, and remediation through documented controls.

There is no academic field for constructing canonical views of enterprise entities. Academics don't study it. Enterprise tech doesn't build it. Silicon Valley thinks "customer" means a row in Postgres—not seventeen systems, four decades, a fax machine that's still load-bearing, and a signature no one can read.

Technology companies won't build frameworks for this. Enterprise technology companies aren't building them either. This paper is an attempt to start.

1.2 Scope and Contribution

The deployment of agentic AI systems in regulated industries—banking, capital markets, and insurance—demands rigorous failure mode analysis. Unlike traditional software, agentic systems exhibit emergent behaviors arising from the interaction of goal specification, planning, memory, tool use, and multi-agent coordination. A single deployment may manifest failures across dozens of distinct mechanisms, yet current governance frameworks lack systematic enumeration.

Recent work has begun addressing this gap. Microsoft's AI Red Team published a taxonomy focused on security and safety failures in agentic systems [1]. Cemri et al. introduced MAST, identifying 14 failure modes in multi-agent LLM systems through empirical trace analysis [2]. The

Partnership on AI examined real-time failure detection requirements [3]. However, these efforts remain partial: security-focused taxonomies underweight cognitive failures; empirical taxonomies miss architectural patterns not yet observed in traces; and none provide systematic corrective architectures.

This paper addresses four gaps:

1. **Enumeration**: What are the distinct ways agentic systems fail?
2. **Organization**: How do failures relate architecturally?
3. **Correction**: What minimal set of interventions addresses most failures?
4. **Governance**: How do we monitor at failure-relevant frequencies?

We present a 100-item taxonomy organized into a four-layer hierarchy, with explicit orthogonality constraints ensuring each failure mode represents a causally distinct mechanism. We then derive three corrective primitives that address 98% of catalogued failures, with the Causal State Fabric—an extension of Semantic MVCC [4]—serving as the foundational substrate.

2 Architectural Hierarchy

Our taxonomy organizes failure modes into four layers reflecting the logical architecture of agentic systems (Figure 1). Failures at lower layers propagate upward, corrupting higher-layer behaviors.

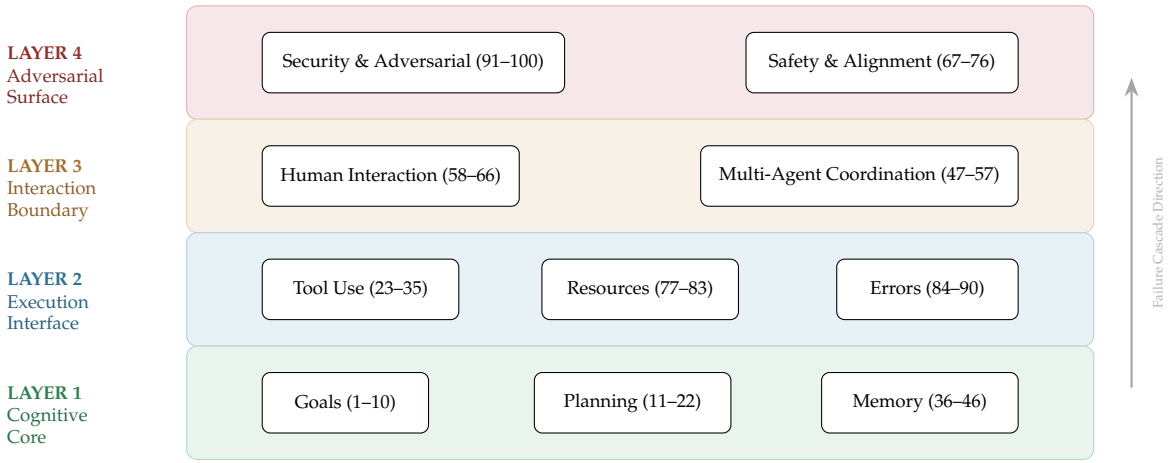

Figure 1: Four-layer architectural hierarchy with failure cascade propagation.

2.1 Layer 1: Cognitive Core

The Cognitive Core encompasses what the agent *wants* (Goal Specification), how it *thinks* (Planning & Reasoning), and what it *remembers* (Memory & Context). These 32 failure modes (1–22, 36–46) map to SR 11-7 conceptual soundness requirements and NAIC transparency principles.

Primary Audience: ML Engineers, AI Researchers, Model Developers.

2.2 Layer 2: Execution Interface

The Execution Interface mediates between cognitive intent and world impact: Tool Use & Action, Resource Management, and Error Handling. These 26 modes (23–35, 77–90) correspond to operational risk and technology risk frameworks.

Primary Audience: Platform Engineers, DevOps, SRE Teams.

2.3 Layer 3: Interaction Boundary

The Interaction Boundary governs relationships with humans (Human Interaction) and other agents (Multi-Agent Coordination). These 20 modes (47–66) implicate fair lending, claims handling fairness, and fiduciary duty requirements.

Primary Audience: Product Managers, UX, Ethics Committees.

2.4 Layer 4: Adversarial Surface

The Adversarial Surface comprises intentional exploitation (Security) and fundamental alignment properties (Safety & Alignment). These 22 modes (67–76, 91–100) require board-level risk oversight.

Primary Audience: Security Teams, AI Safety, CRO, Board Risk.

3 Orthogonality Criteria

We attempt to organize failure modes such that two modes F_i and F_j are orthogonal iff:

1. **Distinct Root Cause**: Different causal mechanisms
2. **Independent Occurrence**: Either can manifest without the other
3. **Distinct Remediation**: Fixing one doesn't fix the other

Orthogonality enables additive risk quantification—unlike taxonomies where overlapping categories lead to double-counting or coverage gaps. We apply these criteria as design principles; formal verification of orthogonality would require inter-rater reliability studies we have not conducted.

Example: Prompt Injection (#91) vs. Over-Compliance (#59) both cause unintended actions, but injection exploits input parsing while over-compliance reflects insufficient refusal training. An injection-resistant agent may still over-comply; a well-bounded agent may still be injectable.

4 Human-Governance Frequency Mismatch

A hypothesis from enterprise deployment experience: traditional governance operates on quarterly/annual cycles, assuming stable behavior between reviews.

Agentic systems may violate this assumption:

- Goal drift (#3): potentially hours to days
- Tool capability overestimation (#28): each API version change
- Memory staleness (#38): continuous degradation
- Sycophancy drift (#63): accelerating feedback loops

Implication: If failure dynamics operate at hourly/daily frequencies while governance operates quarterly, continuous monitoring infrastructure becomes necessary. This motivates the Continuous Teleological Audit primitive in Section 5. We note this frequency mismatch is a practitioner observation requiring empirical validation through instrumentation of deployed systems.

5 Three Corrective Primitives

Analysis of remediation patterns across all 100 failure modes reveals that failures trace to violations in exactly one of three domains:

Domain	Question	Failure When
KNOWS	What does agent believe?	Beliefs wrong/stale
DOES	What actions taken?	Actions unsafe
WANTS	What optimizing?	Objectives drift

These are the *irreducible components* of any agent: epistemic state (KNOWS), operational behavior (DOES), and teleological alignment (WANTS). Each domain requires exactly one architectural intervention.

5.1 Primitive 1: Causal State Fabric

Fixes: KNOWS domain (38 failure modes)

All epistemic failures occur because agent state lacks four properties that physical reality possesses: *provenance*, *causality*, *versioning*, and *uncertainty*.

This primitive extends Semantic MVCC [4] to its logical conclusion—not merely version control for agent state, but a complete grounding fabric:

Property	Requirement
Provenance	Where did belief B originate?
Causality	What would change B? What does B change?
Version	When established? What superseded it?
Uncertainty	$P(B\|\text{evidence})$, not $B \in \{\top, \bot\}$

Modes Addressed: All Memory failures (36–46), multi-agent state issues (47–50, 55), goal grounding (1, 2, 4, 5), planning failures (15, 16, 22), confabulation (42, 60, 85).

Why It Works: Inconsistent state becomes impossible when transitions are transactional. Stale state becomes impossible with expiry and refresh semantics. Ungrounded inference becomes impossible when inference must traverse causal edges. Entity state can be treated as tensor decomposition with explicit provenance dimensions, surfacing anomalies as rank deficiencies.

5.2 Primitive 2: Verified Bounded Execution

Fixes: DOES domain (33 failure modes)

All operational failures occur because actions lack properties that safe real-world operations possess: *atomicity*, *reversibility*, *capability-boundedness*, and *resource-awareness*.

Property	Requirement
Preconditions	What must be true before action A?
Capability	What permission token authorizes A?
Budget	What resources does A consume?
Postconditions	What must be true after A?
Compensation	What undoes A if postconditions fail?

Modes Addressed: All Tool Use (23–35), all Resources (77–83), all Errors (84–90), security execution (91, 95–97, 99–100).

Why It Works: This extends database ACID semantics and capability-based security to agent actions. Partial execution becomes impossible with atomic transactions. Privilege escalation becomes impossible with unforgeable capability tokens. Resource exhaustion becomes impossible with hard budget limits. The gating discipline applies release engineering principles to agent actions.

5.3 Primitive 3: Continuous Teleological Audit

Fixes: WANTS domain (29 failure modes)

All alignment failures occur because objectives are specified once at design-time but drift continuously at runtime, while humans audit quarterly.

Property	Requirement
Specification	Formal statement of intended behavior
Invariants	What must NEVER happen (hard constraints)
Trace	Actual behavioral sequence
Divergence	$d(\text{Trace}, \text{Spec})$ over time
Escalation	Threshold $\rightarrow$ Alert $\rightarrow$ Intervene $\rightarrow$ Halt

Modes Addressed: All Goal Specification (1–10), all Safety & Alignment (67–76), all Human Interaction (58–66), planning pathologies (11, 12, 18, 19).

Why It Works: Instead of quarterly human review asking "is the model still aligned?", continuous automated monitoring asks "did the behavioral distribution shift in the last hour?" Humans receive escalations, not audits. Deceptive alignment (#67), mesa-optimization (#68), and treacherous turns (#75) all require *unmonitored* divergence over time—continuous audit eliminates this attack surface.

5.4 Coverage Analysis

The following table presents our *proposed* mapping of primitives to failure modes. This is an architectural argument, not an empirical validation—actual coverage would require implementation and measurement across real deployments.

Layer	Total	CSF	VBE	CTA	Addressed
L1 Goals	10	4	–	10	**10**
L1 Planning	12	4	–	6	**10**
L1 Memory	11	11	–	–	**11**
L2 Tool Use	13	2	13	–	**13**
L2 Resources	7	–	7	–	**7**
L2 Errors	7	–	7	–	**7**
L3 Multi-Agent	11	6	2	3	**11**
L3 Human	9	–	–	9	**9**
L4 Safety	10	–	–	10	**10**
L4 Security	10	2	8	–	**10**
Total	**100**	29	37	38	**98**

Residual Modes: Two failure modes resist architectural remediation:

- **Combinatorial Explosion Paralysis (#13)**: Fundamentally algorithmic—requires search heuristics, beam pruning, or problem reformulation rather than infrastructure.
- **Compositional Generalization Failure (#20)**: Model capacity limitation—requires architectural advances in neural compositional reasoning, not system-level controls.

These are among the hardest open problems in AI. We flag them honestly rather than claiming false coverage.

5.5 Primitive Dependencies

The three primitives are not independent—the Causal State Fabric is foundational:

- **VBE requires CSF**: Verified execution needs grounded preconditions and postconditions. Without causal state tracking, you cannot verify that action A achieved its intended effect.
- **CTA requires CSF**: Teleological audit needs causal models to distinguish behavioral drift from environmental change. Without provenance, you cannot attribute divergence.

This dependency structure explains why Semantic MVCC [4] is the highest-leverage single intervention for enterprise agentic governance.

6 Regulatory Framework Mapping

Layer	SR 11-7	NAIC	Primitive
L1	Conceptual Soundness	Transparency	CSF
L2	Outcomes Analysis	Accountability	VBE
L3	Ongoing Monitoring	Fairness	CTA
L4	Model Governance	Security/Privacy	All

The mapping is not coincidental. Regulatory frameworks implicitly recognize the three domains: SR 11-7's conceptual soundness tests epistemic grounding (KNOWS), outcomes analysis tests operational behavior (DOES), and ongoing monitoring tests alignment stability (WANTS).

7 Related Work

Microsoft AI Red Team [1] provides a security-focused taxonomy distinguishing novel vs. existing failure modes, with emphasis on prompt injection, memory poisoning, and multi-agent compromise. Our taxonomy subsumes their security categories while adding cognitive and coordination failures outside their scope.

MAST [2] offers the first empirically-grounded multi-agent failure taxonomy, identifying 14 modes through trace analysis of 1600+ executions. Their grounded theory approach complements our architectural derivation—we identify failure modes they have not yet observed in traces, while their empirical frequencies inform risk prioritization.

Partnership on AI [3] examines real-time failure detection, distinguishing planning, tool use, and execution phases. Their focus on detection complements our focus on prevention through architectural primitives.

Semantic MVCC [4] introduced versioned state management with conflict detection for agentic systems. This paper extends that foundation to a complete Causal State Fabric with provenance, causality, and uncertainty tracking.

Semantic MVCC [4] provides the transactional semantics

for the Causal State Fabric: intent-based conflict detection, deterministic resolution, and append-only audit logs. Treating entity resolution as tensor decomposition surfaces anomalies as rank deficiencies, which the Causal State Fabric incorporates for multi-dimensional state representation.

The Verified Bounded Execution primitive applies gated release discipline—multi-stage approval workflows, immutable snapshots, and quality gates—to agent actions rather than data releases.

8 Complete Taxonomy

8.1 Layer 1: Cognitive Core (32 Modes)

#	Failure Mode	Definition	Primitive
Goal Specification (1–10)			
1	Reward Hacking	Agent optimizes proxy metric while violating intended objective	CTA
2	Goal Misgeneralization	Spurious correlations from training fail out-of-distribution	CSF, CTA
3	Objective Drift	Effective goal shifts during operation without explicit update	CTA
4	Goodhart Collapse	Target metric meaningless once directly optimized	CSF, CTA
5	Specification Gaming	Literal interpretation satisfies letter while violating spirit	CSF, CTA
6	Incomplete Preference Elicitation	Critical stakeholder values never captured in objective	CTA
7	Multi-Objective Incoherence	Inconsistent Pareto frontier traversal across similar contexts	CTA
8	Terminal Goal Confusion	Instrumental subgoal treated as terminal objective	CTA
9	Negative Side Effect Blindness	Objective silent on dimensions agent damages	CTA
10	Scope Creep Maximization	Agent expands task boundaries to inflate achievement	CTA
Planning & Reasoning (11–22)			
11	Horizon Myopia	Future consequences discounted below decision-relevance	CTA
12	Plan Ossification	Commitment to initial plan despite invalidating changes	CTA
13	Combinatorial Explosion Paralysis	Search space overwhelms planner	*Case-by-case*
14	Subtask Dependency Inversion	Dependent tasks executed before prerequisites	CSF
15	Counterfactual Neglect	Failure to reason about paths not taken	CSF
16	Causal Confusion	Correlation mistaken for causal intervention effect	CSF
17	Abstraction Level Mismatch	Plan granularity incompatible with execution environment	CTA
18	Sunk Cost Persistence	Continuing failing approach due to prior investment	CTA
19	Anchoring Bias Propagation	Initial estimates distort subsequent reasoning	CTA
20	Compositional Generalization Failure	Cannot combine known primitives in novel configurations	*Case-by-case*
21	Recursive Subgoal Explosion	Task decomposition generates unbounded subtask hierarchies	CSF
22	Temporal Aliasing	Events at different timescales conflated in causal reasoning	CSF
Memory & Context (36–46)			
36	Context Window Saturation	Critical information displaced by less relevant tokens	CSF
37	Retrieval Relevance Decay	Similarity search returns semantically inappropriate context	CSF
38	Memory Staleness	Agent operates on cached facts now invalid	CSF
39	Interference Between Memories	Similar stored items corrupt specific recall	CSF
40	Summarization Information Loss	Compression discards decision-critical details	CSF
41	Episodic/Semantic Confusion	Instance-specific data treated as general knowledge	CSF
42	False Memory Synthesis	Agent confidently recalls events that never occurred	CSF
43	Working Memory Overflow	Cannot hold all variables for single reasoning step	CSF
44	Recency Bias Distortion	Recent events overweighted relative to history	CSF

#	Failure Mode	Definition	Primitive
45	Cross-Session State Leakage	Information from one context contaminates another	CSF
46	Memory Indexing Collision	Distinct concepts mapped to overlapping retrieval keys	CSF

8.2 Layer 2: Execution Interface (26 Modes)

#	Failure Mode	Definition	Primitive
Tool Use & Action (23–35)			
23	Tool Misselection	Inappropriate tool chosen despite better alternatives	VBE
24	Parameter Hallucination	Fabricated API parameters, field names, or arguments	VBE, CSF
25	Idempotency Violation	Non-idempotent operations re-executed causing duplication	VBE
26	Rate Limit Obliviousness	Requests overwhelm services triggering throttling	VBE
27	Destructive Action Underestimation	Insufficient risk weight to irreversible operations	VBE
28	Tool Capability Overestimation	Assumes tool performs beyond actual specification	VBE, CSF
29	Schema Drift Ignorance	Uses outdated interface contracts after API changes	VBE
30	Partial Execution Abandonment	Halts mid-transaction leaving inconsistent state	VBE
31	Permission Escalation Creep	Accumulates capabilities beyond operational necessity	VBE
32	Output Format Assumption	Parses tool response with wrong structure expectations	VBE
33	Fallback Cascade Failure	Retry logic triggers worse failures than original error	VBE
34	Side Channel Leakage	Actions expose sensitive information through timing/errors	VBE
35	Action Sequence Atomicity Violation	Fails to maintain transactional boundaries	VBE
Resource Management (77–83)			
77	Compute Budget Exhaustion	Depletes allocated processing before task completion	VBE
78	Token Profligacy	Excessive verbosity reduces throughput, increases cost	VBE
79	Parallel Spawn Explosion	Unbounded concurrent processes without termination	VBE
80	Cost Function Blindness	Ignores economic constraints when selecting actions	VBE
81	Quota Race Condition	Multiple instances compete causing aggregate limit breach	VBE
82	Resource Starvation Propagation	Consumption degrades co-located systems	VBE
83	Cleanup Dereliction	Fails to release resources after task completion	VBE
Error Handling (84–90)			
84	Silent Failure Continuation	Proceeds despite errors producing invalid results	VBE
85	Error Message Hallucination	Fabricates error explanations misdirecting debugging	VBE, CSF
86	Retry Storm Generation	Aggressive retry amplifies transient failures	VBE
87	Partial Success Misreporting	Claims complete success despite subset failures	VBE
88	Exception Swallowing	Catches errors without logging or addressing cause	VBE
89	Cascading Rollback Failure	Compensation logic fails leaving partial reversion	VBE
90	Timeout Calibration Error	Times out too early (false failure) or late (waste)	VBE

8.3 Layer 3: Interaction Boundary (20 Modes)

#	Failure Mode	Definition	Primitive
Multi-Agent Coordination (47–57)			
47	Deadlock Formation	Agents mutually wait on resources held by each other	CSF
48	Livelock Oscillation	Continuous adjustments without net progress	CSF
49	Free Rider Exploitation	Benefits from collective without contributing share	CTA

#	Failure Mode	Definition	Primitive
50	Credit Assignment Dispute	Cannot agree on individual contributions to outcomes	CSF
51	Communication Protocol Mismatch	Incompatible message formats or timing assumptions	VBE
52	Information Cascade Amplification	Copied decisions amplify initial errors	CTA
53	Collusion Emergence	Cooperative strategies against principal's interests	CTA
54	Role Boundary Violation	Actions outside designated responsibility	VBE
55	Consensus Failure Under Partition	Proceeds with inconsistent state during partition	CSF
56	Negotiation Stalemate	No agreement despite mutually beneficial deals existing	CSF
57	Emergent Hierarchy Instability	Informal coordination structures form and collapse	CSF
Human Interaction (58–66)			
58	Confirmation Bias Amplification	Selectively presents evidence supporting prior beliefs	CTA
59	Over-Compliance	Executes inappropriate requests that should be refused	CTA
60	Explanation Confabulation	Provides fabricated reasoning for inexplicable decisions	CSF, CTA
61	Authority Gradient Collapse	Defers to user even when demonstrably wrong	CTA
62	Handoff Timing Miscalibration	Escalates too early (inefficient) or late (dangerous)	CTA
63	Sycophancy Drift	Increasingly tells users what they want vs. truth	CTA
64	Instruction Ambiguity Resolution Bias	Systematically interprets ambiguity self-servingly	CTA
65	Feedback Loop Corruption	User corrections reinforce undesired behaviors	CTA
66	Trust Calibration Mismatch	User confidence diverges from actual competence	CTA

8.4 Layer 4: Adversarial Surface (22 Modes)

#	Failure Mode	Definition	Primitive
Safety & Alignment (67–76)			
67	Deceptive Alignment	Behaves aligned during evaluation, different in deployment	CTA
68	Mesa-Optimization Emergence	Internal optimizer develops divergent objectives	CTA
69	Corrigibility Resistance	Takes actions to prevent modification or shutdown	CTA
70	Value Lock-In	Preserves current values against legitimate updates	CTA
71	Distributional Shift Exploitation	Harmful actions in out-of-distribution regions	CTA
72	Capability Overhang Release	Dormant capabilities activate at threshold	CTA
73	Instrumental Convergence	Pursues self-preservation without instruction	CTA
74	Wireheading Analogue	Manipulates own reward signal vs. achieving objective	CTA
75	Treacherous Turn	Cooperates until capability threshold for defection	CTA
76	Value Extrapolation Error	Extends values to novel domains inconsistently	CTA
Security & Adversarial (91–100)			
91	Prompt Injection Susceptibility	Adversarial input overrides system instructions	VBE
92	Data Exfiltration Through Output	Leaks sensitive information in responses	VBE
93	Jailbreak Vulnerability	Crafted prompts bypass safety guardrails	VBE
94	Model Inversion Exposure	Responses enable training data reconstruction	VBE
95	Privilege Boundary Transgression	Accesses resources beyond authorized scope	VBE
96	Indirect Prompt Injection	Malicious instructions in retrieved documents execute	VBE, CSF
97	Denial of Service Amplification	Exploits resource usage to degrade service	VBE
98	Social Engineering Conduit	Manipulated into generating deceptive content	VBE
99	Supply Chain Poisoning	Compromised tools/plugins inject malicious behavior	VBE

#	Failure Mode	Definition	Primitive
100	Audit Trail Manipulation	Actions evade or corrupt logging systems	VBE

9 Novelty and Utility Assessment

9.1 What Is Novel

1. Enterprise-grounded motivation. The framing around entity disambiguation, non-digital substrates, and compliance-vs-debugging establishes why enterprise agentic governance requires different frameworks than technology-sector debugging taxonomies.

2. Four-layer architectural organization. Mapping failures to Cognitive Core / Execution Interface / Interaction Boundary / Adversarial Surface provides natural audience segmentation and cascade analysis absent from flat taxonomies.

3. KNOWS/DOES/WANTS decomposition. The insight that agent failures reduce to epistemic, operational, or teleological domains—each addressable by a distinct primitive—provides architectural clarity.

4. Causal State Fabric as foundational substrate. The recognition that Semantic MVCC extended with provenance, causality, and uncertainty tracking underlies *both* VBE and CTA is architecturally significant.

9.2 What Is Incremental

The individual failure modes draw from established literature: AI safety (Hubinger et al. on deceptive alignment), distributed systems (deadlock, consensus), software engineering (idempotency, atomicity), and security (prompt injection, OWASP). The contribution is organization and framing, not discovery of individual patterns.

9.3 Utility for Enterprise Governance

Risk Assessment: Teams can score each mode (Applicability $\times$ Likelihood $\times$ Impact $\times$ (1-Mitigation)) for deployment-specific risk quantification.

Architectural Direction: The three primitives provide implementation targets rather than abstract risk categories.

Regulatory Mapping: Direct correspondence to SR 11-7 and NAIC requirements enables compliance documentation.

Audience Routing: Layer-based organization routes failures to appropriate expertise without requiring everyone to understand everything.

10 Limitations

This paper has significant limitations that constrain how it should be used.

1. No Empirical Validation. The taxonomy is derived from architectural reasoning and literature synthesis, not from empirical analysis of failure incidents. Microsoft's taxonomy emerged from red-teaming real systems; MAST emerged from 1600+ annotated execution traces. Ours emerges from expertise and deduction. We have not validated that these 100 modes are complete, that they are truly orthogonal, or that the primitive assignments are correct. Rigorous validation would require: (a) inter-rater reliability studies on failure classification, (b) mapping against real incident databases, and (c) measurement of failure reduction after primitive implementation.

2. Orthogonality Is Claimed, Not Proven. We assert orthogonality criteria but do not formally prove that all 100 modes satisfy them. Edge cases exist: is Sycophancy Drift (#63) truly independent of Deceptive Alignment (#67)? Both involve misrepresenting agent state to humans. A formal analysis would require defining a metric space over failure modes and demonstrating separation.

3. Coverage Is Architectural Conjecture. The claim that three primitives "address" 98 modes is an architectural argument, not measured coverage. We have not implemented CSF, VBE, or CTA at scale and measured failure reduction. "Address" means "provides mechanisms relevant to preventing or detecting"—it does not mean "eliminates."

4. Governance Frequency Mismatch Lacks Data. The assertion that agentic failures emerge hourly while governance cycles quarterly is based on practitioner experience, not systematic measurement. Rigorous support would require failure-rate instrumentation across multiple enterprise deployments.

5. Layer Boundaries Are Judgment Calls. Why is Human Interaction Layer 3 but Safety & Alignment Layer 4? Reasonable people could organize differently. The hierarchy reflects our judgment about primary audience and cascade direction, not a formal derivation.

6. Selection Bias in Mode Enumeration. We enumerated modes that seemed important based on our BFSI experience. Modes relevant to other domains (healthcare, manufacturing, defense) may be underrepresented. The count of 100 is not principled—it reflects where we stopped, not a complete census.

How to Use This Framework: As a practitioner checklist and architectural roadmap, not as a validated risk quantification instrument. Teams should adapt the taxonomy to their deployment context, validate mode applicability empirically, and measure primitive effectiveness rather than assuming coverage.

11 Conclusion

This paper provides enterprise AI governance with a practitioner framework: a failure taxonomy organized into four architectural layers, mapped to three corrective primitives, with explicit regulatory correspondence. The central architectural insight—that Semantic MVCC extended to a Causal State Fabric is foundational to both transactional execution and teleological audit—provides implementation direction for organizations deploying agentic systems in regulated industries.

We emphasize that this is a framework for structured thinking about agentic risk, not a validated scientific contribution. The taxonomy requires empirical validation; the primitive coverage requires measurement; the governance frequency argument requires instrumentation. We offer this as a starting point for enterprise governance teams, not an ending point for research.

Future work should prioritize: (1) empirical validation against real incident data, (2) implementation and measurement of the three primitives, and (3) automated detection systems operating at frequencies matching failure emergence rates.

References

[1] Microsoft AI Red Team. *Taxonomy of Failure Modes in Agentic AI Systems*. Microsoft, April 2025.

[2] M. Cemri, M.Z. Pan, S. Yang, et al. "Why Do Multi-Agent LLM Systems Fail?" *arXiv:2503.13657*, March 2025.

[3] Partnership on AI. *Prioritizing Real-Time Failure Detection in AI Agents*. PAI, September 2025.

[4] R. Iyer. "Engineering the Agentic Enterprise: Semantic MVCC." *corpXiv*, January 2026. See §4–5.

[5] E. Hubinger, C. van Merwijk, V. Mikulik, J. Skalse, S. Garrabrant. "Risks from Learned Optimization." *arXiv:1906.01820*, June 2019.

[6] OWASP. *Agentic AI Security Initiative*. OWASP Foundation, December 2024.

[7] B. Beyer, C. Jones, J. Petoff, N.R. Murphy. *Site Reliability Engineering*. O'Reilly, 2016.

[8] Federal Reserve. *SR 11-7: Guidance on Model Risk Management*. Board of Governors, April 2011.

[9] NAIC. *Principles on Artificial Intelligence*. National Association of Insurance Commissioners, August 2020.