The BX3 Framework: A Universal Architecture for Accountable Autonomous Systems

Jeremy Blaine Thompson Beebe
Bxthre3 Inc. — bxthre3inc@gmail.com — ORCID: 0009-0009-2394-9714

April 2026

Abstract

The current discourse around artificial intelligence presents a false binary: either AI replaces traditional software, or it is a limited novelty. Both fail engineers, architects, and decision-makers by offering no principled model for how these technologies relate. This paper proposes the **BX3**Framework — a universal architectural model organized around three immutable functional layers: the *Purpose Layer*, which provides intent, judgment, and accountability; the *Bounds Engine*, which provides bounded reasoning and constrained proposal; and the *Fact Layer*, which provides deterministic enforcement and forensic auditability. Critically, the framework is actor-agnostic: any entity — human, AI, mechanical, or institutional — may occupy any layer provided it satisfies the layer's functional requirements. The framework's core safety property is an upstream accountability guarantee: when any node encounters a condition it cannot resolve, accountability escalates recursively upward through the hierarchy, bypassing machine actors, until it reaches a human anchor. The system fails upward into human consciousness. It never fails downward into algorithmic chaos. The **BX3**Framework is demonstrated as a universal structural pattern observable across law, medicine, autonomous vehicles, organizational design, and biological cognition.

Keywords: BX3 Framework, Purpose Layer, Bounds Engine, Fact Layer, autonomous systems, AI governance, human-in-the-loop, deterministic systems, sociotechnical systems, accountability, recursive orchestration

1 Introduction

The rapid deployment of large language models and AI-based systems has generated significant confusion about the appropriate role of AI within engineered systems. A prevalent narrative suggests AI will progressively replace traditional deterministic software. A counter-narrative dismisses AI as inadequate for production reliability demands. Both positions offer no principled model for how these technologies relate in practice.

This paper proposes the **BX3**Framework. The framework organizes any complex system into three immutable functional layers — *Purpose Layer*, *Bounds Engine*, and *Fact Layer*— each defined by the properties it must maintain rather than by the type of actor occupying it.

This actor-agnostic definition is the framework's most important architectural property. A human, an AI system, a mechanical process, an institution, or any combination thereof may legitimately occupy any layer, provided the functional requirements of that layer are satisfied. The framework does not prescribe *who* belongs in each layer but *what properties* each layer must maintain for the system as a whole to be reliable, governable, and certifiable.

The framework synthesizes well-established principles — separation of concerns [**?**], sociotechnical systems theory [**?**], control theory [**?**], and human-in-the-loop design [**?**] — applied to the specific challenge of AI integration in the current technological moment.

Note: The first-person plural "we" is used in the conventional academic sense, consistent with single-author scholarly writing.

2 Defining the Three Functional Layers

The **BX3**Framework defines three functional layers, each defined by the *properties it must maintain*, not by the type of actor occupying it.

2.1 Layer 1: Purpose Layer

The Purpose Layer is responsible for *intent*, *judgment*, and *accountability*. Whatever occupies this layer must be capable of defining the goals the system exists to achieve, making trade-off decisions under genuine uncertainty where no algorithmic optimum exists, holding accountability for outcomes, and updating goals when context changes in ways the system was not designed to anticipate.

In the current technological moment, the Purpose Layer must remain anchored to a *human accountability anchor* — an individual or institution capable of bearing legal and ethical responsibility. This is the **Human Root Mandate**: in the event of system failure, accountability does not dissipate into the algorithm. It remains fixed to the human at the root.

Key property: The Purpose Layer must be *accountable* — its decisions must be attributable to an actor who can be questioned, overridden, and held responsible.

2.2 Layer 2: Bounds Engine

The Bounds Engine is responsible for *interpretation*, *bounded reasoning*, and *constrained execution*. It performs cognitive work — analysis, pattern recognition, simulation, and optimal path proposal — but is architecturally *limbless*: it can propose but cannot execute. It lacks authority to commit actions to the physical world unilaterally.

This layer is most commonly occupied by an AI agent. It operates within a sandboxed cognitive environment, separated from physical execution authority, and escalates to the Purpose Layer when situations exceed its capability or authority.

Key property: The Bounds Engine must be *bounded* — it proposes but never executes; its authority is constrained by the Safety Envelope.

2.3 Layer 3: Fact Layer

The Fact Layer is responsible for *deterministic enforcement*, *hard physical constraint*, and *forensic auditability*. Whatever occupies this layer must produce consistent, reproducible outcomes given identical inputs, enforce physical constraints that cannot be overridden by the Bounds Engine, and maintain an immutable record of all physical events.

The Fact Layer is the only layer that can act on the physical world. It is structurally incapable of reasoning — it executes, it does not deliberate.

Key property: The Fact Layer must be *deterministic* — same inputs, same outputs, every time, without exception.

2.4 Layer Comparison

Property	**Purpose Layer**	**Bounds Engine**	**Fact Layer**
Function	Intent, judgment, accountability	Reasoning, proposal, simulation	Enforcement, constraint, audit
Default occupant	Human / institution	AI agent / heuristic	Software / mechanism
Key obligation	Accountability anchor	Boundedness (limbless)	Determinism (no drift)

3 The Five Pillars

The three functional layers define *what* a **BX3**-compliant system must contain. The Five Pillars define *how* those layers must behave in practice to maintain their properties under real-world conditions including network failures, scale, security threats, and exception states.

3.1 Pillar 1: Loop Isolation

Problem solved: Logic Collision — when the Bounds Engine and Fact Layer occupy the same functional plane, enabling un-vetted autonomous actions that bypass physical constraint.

Solution: Strict isolation of the three functional layers into discrete planes. Each **BX3**loop is self-contained. A Logic Collision is architecturally impossible because the Bounds Engine never shares a functional plane with physical execution. The Bounds Engine proposes; the Fact Layer decides. These are never the same operation.

3.2 Pillar 2: Recursive Spawning

Problem solved: Logic Rigidity — static edge devices that cannot adapt to local conditions without constant cloud connectivity.

Solution: A parent Bounds Engine births a child loop by generating a *Worksheet* — a containerized, self-contained logic set encapsulating the parent's Purpose for a specific local context — deployed over-the-air to the child node. Each Worksheet carries a hard-coded pointer to the parent's Purpose, preventing autonomous drift. The child loop applies the parent's intent to local sensor data independently without requiring a constant cloud heartbeat.

3.3 Pillar 3: Spatial Firewall

Problem solved: Cross-tier data leakage — a compromised or unauthorized node accessing data at a resolution or scope beyond its provisioned tier.

Solution: Physical isolation of data planes by resolution tier. A node provisioned at 50-meter resolution cannot access 1-meter data because the 1-meter data plane does not exist in the node's provisioned environment. Isolation is enforced by architecture, not by permissions.

3.4 Pillar 4: Sandbox Gate

Problem solved: Premature execution — the Bounds Engine proposing actions that, if executed, would violate Safety Envelope parameters before the system has an opportunity to evaluate them.

Solution: All proposed actions are evaluated in a digital twin before physical execution. The Sandbox Gate runs the proposed action against a simulation of the Fact Layer's current state and confirms it falls within all Safety Envelope parameters before unlocking physical execution.

3.5 Pillar 5: Bailout Protocol

Problem solved: Accountability orphaning — an autonomous system encountering a condition it cannot handle and either halting without notification or continuing without authorization.

Solution: Every node carries the complete Bailout Protocol. When a node encounters a condition it cannot resolve, it propagates an exception asynchronously up the recursive tree, bypassing all intermediate machine actors, until it reaches a Human Accountability Anchor. The system fails upward into human consciousness.

4 Applications

The **BX3**Framework is not merely a software engineering principle. Its three-layer structure is a universal structural pattern observable across many complex domains.

4.1 Autonomous Systems

In sensor-driven autonomous systems, human engineers occupy the Purpose Layer defining mission parameters and safety constraints. AI processing layers occupy the Bounds Engine interpreting sensor data and proposing decisions within constraints. Deterministic control systems occupy the Fact Layer enforcing hard constraints — collision avoidance thresholds, actuator limits, safety interlocks — that cannot be overridden by the Bounds Engine.

4.2 Legal Systems

Legislatures and judges occupy the Purpose Layer — defining law, exercising judgment, bearing institutional accountability. AI systems increasingly occupy the Bounds Engine — assisting with legal research and pattern recognition. The written law, procedural rules, and enforcement mechanisms occupy the Fact Layer — applying consistently regardless of parties involved.

4.3 Medicine

Physicians occupy the Purpose Layer — exercising judgment incorporating patient context and ethical considerations. AI systems occupy portions of the Bounds Engine — imaging analysis, drug interaction checking. Drug dosage protocols, surgical checklists, equipment specifications, and regulatory requirements occupy the Fact Layer — deterministic constraints binding both physician and AI.

4.4 Biological Cognition

The **BX3**Framework mirrors the layered architecture of biological cognition [**?**]. The autonomic nervous system and reflexes occupy the Fact Layer — deterministic, fast, non-negotiable. Learned heuristics and pattern recognition occupy the Bounds Engine — efficient responses to familiar situations. Conscious deliberative reasoning occupies the Purpose Layer — slow, deliberate, engaged only for genuinely novel high-stakes situations.

5 Why Determinism Is Not Compensable

Determinism provides properties that probabilistic systems cannot replicate:

- **Reproducibility:** Same inputs produce same outputs, enabling debugging, auditing, and legal accountability.
- **Formal verification:** Mathematical proof that a system satisfies properties under all possible inputs.

- **Certification:** Regulatory frameworks in aviation, medical devices, and safety-critical infrastructure require deterministic behavior as a precondition for approval.
- **Latency:** Hard real-time requirements are achievable with deterministic systems and not with current AI inference pipelines.

Every AI system in production today runs on top of vast deterministic infrastructure: operating systems, databases, networking stacks, authentication systems. The claim that AI will replace deterministic software is structurally self-refuting — AI systems are themselves dependent on deterministic software.

6 Related Work

The **BX3**Framework synthesizes established ideas from separation of concerns, sociotechnical systems theory, cybernetics, systems safety, and human-in-the-loop design, but departs from prior work by defining immutable functional layers according to required properties rather than actor type.

The intelligent sociotechnical systems framework [**?**] similarly emphasizes structured coordination between human and technical components. **BX3**extends this by explicitly isolating a deterministic Fact Layer and specifying an actor-agnostic architecture where any occupant of a layer is bound by that layer's functional obligations.

Tiered Agentic Oversight (TAO) [**?**] demonstrates that hierarchical supervision among specialized agents reduces error propagation. **BX3**differs by making human accountability the required terminal endpoint of unresolved escalation rather than a high-tier supervisory option.

Recent work on production-grade agent architectures [**?**] recommends fail-safe behavior, sandbox-first execution, and human approval gates. **BX3**incorporates these as named architectural pillars rather than implementation heuristics.

BX3 sits naturally alongside ISO/IEC 42001 [**?**] and the NIST AI RMF [**?**], which require translation across governance, design-time, runtime, and assurance layers. **BX3**provides one candidate architecture for that translation.

7 Conclusion

The question of how humans, AI, and traditional software should relate is an architectural, organizational, ethical, and regulatory question with significant practical consequences.

The **BX3**Framework proposes a principled, universal answer: organize any complex system into three functional layers — Purpose, Bounds Engine, and Fact — each defined by the properties it must maintain rather than the actor type occupying it. Any actor capable of satisfying a layer's functional requirements may occupy that layer: human, AI, mechanical, institutional, or hybrid.

This actor-agnostic definition is the framework's most important contribution. It makes **BX3**applicable to human-only organizations, fully automated pipelines, multi-agent AI architectures, and hybrid compositions that do not yet have names. In each case, the framework asks the same three questions: Is there an accountable layer? Is there a bounded layer that

reasons and proposes? Is there a deterministic layer that enforces and audits? If any layer is missing, the system is architecturally incomplete — regardless of how capable its components are individually.

The pattern is visible in legal systems, medical practice, autonomous vehicles, biological cognition, and organizational design — wherever reliable systems have been built to handle a world that is simultaneously rule-bound and unpredictable. What is new is the urgency of making the pattern explicit at a moment when the temptation to collapse these roles, and the cost of doing so, have never been higher.

Acknowledgments

The author acknowledges the foundational contributions of researchers cited herein, whose work across control theory, sociotechnical systems, and computer science provides the foundation for this synthesis.