The BX3 Framework:

Implementation Protocols for Accountable Autonomous Systems

Three Functional Layers. Five Named Protocols. Guaranteed Upstream Accountability.

Jeremy Blaine Thompson Beebe
Independent Researcher
April 2026

Abstract

The current discourse around artificial intelligence frequently presents a false binary: either AI will replace traditional software systems, or it is merely a productivity tool of limited consequence. This paper argues that both positions miss a more fundamental and practically useful insight. We propose 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 interpretation, bounded reasoning, and constrained execution; and the *Fact Layer*, which provides deterministic enforcement, hard physical constraint, and forensic auditability.

Critically, the BX3 Framework does not prescribe *who* or *what* occupies each layer. It prescribes the *functional properties* each layer must maintain — and holds any actor occupying that layer to those properties regardless of their nature. A human, an AI system, or a mechanical process may each legitimately occupy any layer, provided they satisfy that layer's functional requirements. This actor-agnostic, function-centered definition makes the framework universally applicable across human organizations, fully automated systems, multi-agent AI architectures, and any hybrid composition.

A core safety property of the framework is its upstream accountability guarantee: when any node encounters a state it cannot resolve within its bounds, accountability escalates recursively upward through the system hierarchy — bypassing all machine actors — until it reaches a human anchor. *The system fails upward into human consciousness — never downward into algorithmic chaos.*

We demonstrate that when these three functional layers are clearly separated and their properties enforced by design, systems become simultaneously more capable and less complex. We further argue that the Bounds Engine is most powerfully employed

not as a replacement for the Fact Layer but as a tool for designing, accelerating, and improving it. We show that the BX3 Framework is not merely a software engineering principle but a universal structural pattern observable across law, medicine, autonomous systems, organizational design, and biological cognition.

Keywords: BX3 Framework, Purpose Layer, Bounds Engine, Fact Layer, artificial intelligence, deterministic systems, software architecture, human-in-the-loop, upstream accountability, autonomous systems, AI governance

---

1 Introduction

The rapid advancement and 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 that AI will progressively replace traditional deterministic software, eventually subsuming most computational tasks. A counter-narrative dismisses AI as a novelty, inadequate for the reliability demands of production systems.

Both narratives fail engineers, architects, and decision-makers in the same fundamental way: they offer no principled model for how these technologies relate to one another or how they should be combined in practice. This paper proposes a clarifying framework: the **BX3 Framework**. The framework organizes any complex system into three immutable functional layers — the *Purpose Layer*, the *Bounds Engine*, and the *Fact Layer* — each defined by the properties it must maintain rather than by the type of actor that occupies it.

This actor-agnostic, function-centered definition is the framework's most important architectural property. A human, an AI system, or a mechanical process may each legitimately occupy any layer of the BX3 Framework — provided they satisfy the functional requirements of that role. What the framework prescribes is not *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. Its value lies not in the novelty of its components but in the clarity, universality, and completeness of their unification.

2 Defining the Three Functional Layers

The BX3 Framework defines three functional layers. Each layer is defined by the *properties it must maintain*, not by the type of actor that occupies it.

2.1 Layer 1: The Purpose Layer

The Purpose Layer is responsible for *intent*, *judgment*, and *accountability*. It sets Service Level Objectives, strategic goals, and the "why" that governs all downstream activity. Whatever occupies this layer must be capable of: defining the goals the system exists to achieve; making trade-off decisions under genuine uncertainty; 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 for the system's actions. 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.

2.2 Layer 2: The Bounds Engine

The Bounds Engine is responsible for *interpretation*, *bounded reasoning*, and *constrained execution*. It performs the cognitive work of the system — analysis, pattern recognition, simulation, and optimal path proposal — but is architecturally *limbless*: it can propose but cannot execute. Whatever occupies this layer must be capable of: receiving goals and constraints from the Purpose Layer and translating them into proposed actions; handling ambiguous inputs; performing complex analysis within defined boundaries; and escalating to the Purpose Layer when situations exceed its authority.

2.3 Layer 3: The Fact Layer

The Fact Layer is responsible for *deterministic enforcement*, *hard physical constraint*, and *forensic auditability*. It operates on the principle that certain outcomes are not matters of opinion or probability: a valve is open or closed; a water allocation is consumed or available; a sensor reading is recorded or missing. Whatever occupies this layer must be capable of: deterministic execution of validated actions; enforcement of physical and regulatory constraints; tamper-evident logging of all outcomes; and rejection of any proposed action that violates Safety Envelope parameters.

3 The Five Pillars of BX3 Implementation

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.

3.1 Pillar 1: Loop Isolation

Problem solved: Logic Collision — when the Bounds Engine and the 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 and operates independently. A Logic Collision is architecturally impossible because the Bounds Engine never shares a functional plane with physical execution.

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 node births a child BX3 loop by generating a *Worksheet* — a containerized, self-contained logic set encapsulating the parent's Purpose for a specific local context — and deploying it over-the-air to the child node. Each Worksheet carries a hard-coded pointer to the parent's Purpose, preventing autonomous drift.

3.3 Pillar 3: Spatial Firewall

Problem solved: Data access is treated as a software permission, enforced by code that can be bypassed.

Solution: Resolution is treated as a physical property of the Fact Layer — enforced not by software permissions but by the structure of the data architecture itself. A node provisioned at a given resolution cannot access data at higher resolution because it does not exist in that node's data plane.

3.4 Pillar 4: Sandbox Gate

Problem solved: Proposed actions are executed without pre-execution validation against Safety Envelope parameters.

Solution: Every proposed action is validated in a sandbox before execution. The Fact Layer runs a simulation of the proposed action against all Safety Envelope parameters. Only actions that pass all validation checks proceed to execution.

3.5 Pillar 5: Bailout Protocol

Problem solved: Exceptions are handled by static error-handling code that cannot anticipate novel failure modes.

Solution: When any node encounters a condition it cannot resolve, it propagates the exception upward through the parent chain until it reaches a Human Accountability Anchor. Machine actors are excluded from the accountability chain — the propagation bypasses all machine nodes.

4 Formal Specification

The BX3 Framework can be formalized as follows. Let a system S be BX3-compliant if and only if S contains three non-overlapping functional layers L_1, L_2, L_3 such that:

Postulate 1 (Layer Separation): $L_1 \cap L_2 \cap L_3 = \emptyset$, and no layer may write to any other layer.

Postulate 2 (Purpose Accountability): All decisions originating in L_1 are attributable to a named human entity. L_1 decisions cannot be attributed to an AI or software system alone.

Postulate 3 (Bounds Limblessness): L_2 can propose actions but cannot execute them. L_2 has no direct access to physical actuators or databases outside its own reasoning state.

Postulate 4 (Fact Determinism): Given identical inputs and identical L_1 authorization, L_3 produces identical outputs. L_3 is deterministic regardless of L_2 state.

Postulate 5 (Upstream Accountability): Any state $s \in S$ that cannot be resolved by L_2 within Safety Envelope parameters escalates to L_1. Escalation is mandatory and cannot be suppressed by any L_2 node.

5 Comparative Analysis

The BX3 Framework differs from existing approaches in several key dimensions:

Property	BX3	ROS/Kubernetes	TAO
Layer separation	Mandatory	Partial	Optional
Human root mandate	Architectural	None	High-risk path
Deterministic fact layer	Yes	No	No
Upstream accountability	Guaranteed	None	Optional

6 Universal Analogs

The three-layer functional architecture of the BX3 Framework appears across many complex domains where different types of actors occupy each layer:

- **Autonomous systems:** Human engineers define mission parameters (L_1); AI systems propose decisions (L_2); deterministic control systems enforce physical constraints (L_3).
- **Legal systems:** Legislatures and judges occupy L_1; AI assists with legal research in L_2; written law and procedural rules occupy L_3.
- **Medicine:** Physicians exercise judgment in L_1; AI assists with diagnosis and analysis in L_2; drug protocols and safety interlocks occupy L_3.
- **Biological cognition:** The autonomic nervous system occupies L_3; learned heuristics occupy L_2; conscious deliberation occupies L_1.

7 Why Deterministic Systems Will Not Be Replaced

A common claim is that sufficiently advanced AI will eventually subsume deterministic software. We argue this position misunderstands the distinct value of determinism as a *property*, not merely a *limitation*. Determinism provides reproducibility, formal verification, certification for safety-critical infrastructure, and hard real-time latency guarantees that probabilistic AI systems cannot match.

Every AI system in production today runs on top of vast deterministic infrastructure. The claim that AI will replace software is structurally self-refuting — the AI systems making this possible are themselves dependent on deterministic software that will not and should not be replaced.

8 Relationship to Prior Work

The BX3 Framework synthesizes ideas from separation of concerns [**?**], sociotechnical systems theory [**?**], control theory [**?**], human-in-the-loop design [**?**], and agentic AI governance [**?** **?** **?**]. It departs from prior work by defining immutable functional layers according to required properties rather than actor type, and by making human accountability the unconditional terminal endpoint of unresolved escalation rather than an optional high-tier pathway.

9 Conclusion

The BX3 Framework organizes any complex system into three functional layers — Purpose, Bounds Engine, and Fact — each defined by the properties it must maintain rather than by the type of actor that occupies it. Any actor capable of satisfying a layer's functional requirements may occupy that layer: human, AI, mechanical, institutional, or hybrid. What cannot vary are the properties themselves — accountability in the Purpose Layer, boundedness in the Bounds Engine, and determinism in the Fact Layer.

The framework asks the same three questions for any system: Is there an accountable layer that sets purpose? Is there a bounded layer that reasons and proposes within constraints? Is there a deterministic layer that enforces and audits? If any layer is missing or its properties are not maintained, the system is architecturally incomplete — regardless of how capable its components are individually.

Its 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, naming its layers by function rather than actor, and building from it deliberately 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 the researchers cited herein, whose work across control theory, sociotechnical systems, and computer science provides the shoulders on which this synthesis stands.

This work has not undergone peer review. Comments and correspondence are welcomed.