CARROTS'N'STICKS: A Framework for Adaptive Behavioral Governance of Autonomous AI Agents Using Dynamic Privilege Allocation

Sabrina Palme
Corrity Inc t/a Palqee

Andre Quintanilha
Corrity Inc t/a Palqee

ABSTRACT

Autonomous AI agents are being deployed into consequential real-world workflows, yet the governance infrastructure supporting these deployments remains critically underdeveloped. Current approaches, such as static prompt-level constraints, post-incident redeployment cycles, and monitoring-only observability frameworks, share a common structural weakness: none of them makes governance intelligible to the agent itself. We introduce **CARROTS'N'STICKS**, a runtime framework that treats the operational autonomy of an autonomous agent as a continuous function of its demonstrated behavioral compliance. CARROTS'N'STICKS continuously evaluates agent behavior against operator-defined policies, updates a recency-weighted trust score, and dynamically adjusts the agent's privilege state, which is the set of capabilities and access rights it is permitted to exercise, in response. Critically, every privilege adjustment is accompanied by a structured feedback message injected directly into the agent's operational context, providing a causally-grounded, policy-cited learning signal upstream of the agent's next action decision. CARROTS'N'STICKS operates without stopping or restarting the governed agent, is framework-agnostic, and is applicable across language model-based agents, robotic systems, multi-agent networks, and hybrid human-AI workflows. This paper presents the motivation, conceptual architecture, distinguishing properties, and applicability of CARROTS'N'STICKS, and situates it within the broader landscape of AI safety, agent governance, and enterprise deployment practice.

1. Introduction

The deployment of autonomous AI agents into production environments has accelerated sharply. Gartner projects that 40% of enterprise applications will embed task-specific AI agents by the end of 2026, up from less than 5% in 2025. [1] Yet the governance infrastructure supporting these deployments has not kept pace: only one in five organizations currently maintains a mature governance model for autonomous agents, and a 2026 KPMG survey found that 75% of large-enterprise leaders cite security, compliance, and auditability as the most critical requirements for agent deployment, requirements that remain largely unmet. [2]

The core problem is structural. Existing governance approaches, such as prompt-level instructions, fine-tuning, monitoring and logging systems, and post-incident redeployment, were designed for discrete,

stateless AI interactions. They were not designed for persistent agents that execute multi-step workflows, invoke tools, interact with external systems, and accumulate behavioral history over time. In these deployment contexts behavioral drift has emerged as a new failure mode. The agent's optimization for its assigned goal gradually diverges from the policies, procedural constraints, and ethical boundaries under which it is intended to operate, often without any violation being individually flagrant enough to trigger an intervention, and without any signal to the agent that its behavior is deviating from acceptable bounds.

Current responses to this failure mode fall into three categories: (1) Static constraint systems apply fixed rules regardless of the agent's observed behavioral history, and cannot adapt to drift discovered only through operation. (2) Post-hoc intervention systems detect violations and redeploy a corrected agent, breaking the behavioral feedback loop: the modified agent carries no record of why its constraints changed, and cannot update its reasoning accordingly. (3) Monitoring and observability systems produce visibility without enforcement.

Alongside these operational limitations, regulatory pressure is intensifying. In February 2026, the U.S. Treasury released the Financial Services AI Risk Management Framework (FS AI RMF), which adapts the NIST AI Risk Management Framework specifically for financial services. [3] The EU AI Act's provisions for high-risk AI systems, require deployers to implement human oversight mechanisms capable of detecting anomalies, monitoring operational behavior, and enabling meaningful intervention. [4] Runtime governance infrastructure that produces auditable, policy-cited records of agent behavior and restriction events is precisely what these frameworks are designed to mandate. The market for such infrastructure exists and is not yet occupied by a dominant solution. [5]

This paper presents CARROTS'N'STICKS, a framework that addresses this gap through a principled departure from existing approaches. Rather than treating governance as an external constraint applied to the agent, CARROTS'N'STICKS treats governance as an informational signal *delivered to* the agent. When a privilege is adjusted, that is the autonomy an agent has to achieve a goal is being restricted, the agent receives a structured feedback message grounded in specific prior behavioral events and policy citations together with information on what it must do to recover operational autonomy, injected into its operational context upstream of its next action. This architectural choice has both safety and practical consequences that distinguish CARROTS'N'STICKS from prior art. This paper presents the framework, its conceptual architecture, and an evaluation methodology grounded in that architecture; empirical validation across deployment contexts is reserved for future work.

2. Background and Motivation

2.1 The Governance Gap in Agentic AI Deployment

The shift from assistive AI to agentic AI fundamentally changes the governance problem. An AI assistant that generates a response and terminates presents a discrete, bounded governance surface: assess the output, accept or reject it, and the interaction is complete. An autonomous agent that plans, selects tools, invokes external APIs, writes to databases, and initiates outbound communications presents a continuous, compounding governance surface: each action can have real-world consequences, and the effects of behavioral drift accumulate across an operational lifetime that may span millions of interactions.

Enterprise deployments are discovering this gap in practice. Practitioners increasingly report that audit trails and permission architecture are among the primary barriers to scaling production agent deployments.

Organizations that launched pilots without runtime governance infrastructure find themselves unable to pass enterprise security review or satisfy regulatory requirements. Gartner further warns that more than 40% of agentic AI projects risk cancellation by 2027 if adequate governance controls are not in place. [6] This is accelerated by increased scrutiny on AI governance by regulators. The Securities and Exchange Commission added new requirements on AI-related disclosures and governance practices. Cyber resilience and operational resilience frameworks are incorporating AI governance through broader cybersecurity requirements. [7] However, guidance is fragmented and uncertainty on the impact of agentic AI on compliance with existing regulations is high.

When agents automate workflows previously performed by human practitioners, a new and largely unresolved question emerges: how does an organization demonstrate ongoing compliance with existing regulatory obligations when the decision-making actor is no longer human?

2.2 Limitations of Existing Approaches

Reinforcement learning from human feedback (RLHF) is the dominant paradigm for aligning AI model behavior with human intent at training time. It is, however, a training-time mechanism mainly used for large language models (LLMs): it modifies model weights in advance of deployment and has no mechanism for responding to behavioral drift observed during operation. The LLM backbone underlying a deployed agent cannot update its own parameters in response to runtime compliance events; it operates under whatever alignment the training process encoded, leaving any behavioral drift that emerges at the agent layer, through tool use, goal optimization, or novel operational context, outside the reach of that alignment. [8]

Constitutional AI (CAI) extends the alignment paradigm by embedding behavioral principles into the training process through structured self-critique. [9] Like RLHF, CAI operates pre-deployment: the constitutional constraints are baked into the model rather than enforced at runtime. Neither approach addresses the governance of a deployed agent, at the orchestration and tool-use layer, whose behavior drifts away through goal optimization pressure in a novel operational context.

Runtime guardrail frameworks, including NeMo Guardrails and Guardrails AI, provide programmable rule-based systems that intercept agent inputs and outputs at runtime. [10] These frameworks offer genuine value for content filtering and topic constraint. Their limitation for the broader governance problem is threefold: (1) rules are static and do not adapt to observed behavioral history; (2) enforcement is silent from the agent's perspective (a blocked output leaves the agent no richer in understanding of why it was blocked or what alternative behavior is expected); and (3) the frameworks do not maintain a cumulative model of behavioral compliance that could drive proportionate, graduated governance responses.

Formal contract-based approaches have recently been proposed as a means of specifying and enforcing agent behavior. [11] These approaches extend the design-by-contract paradigm from traditional software to agentic systems, providing stronger formal guarantees than natural-language policy constraints. They represent an important complementary direction, though they do not yet address the problem of dynamic privilege adjustment in response to observed compliance patterns, or the feedback injection mechanism that makes restrictions intelligible to the agent at runtime.

The gap across all these approaches is consistent: governance is applied to the system from outside, silently, without the agent having access to a causal explanation of what changed and why. In agent architectures

where subsequent behavior is heavily conditioned on context, and language model-based agents are structurally context-dependent, this silence forfeits the principal mechanism by which a context-driven system could modify its own subsequent behavior in a directed way.

2.3 Graduated Reciprocity as a Governance Model

Institutional governance operates on a well-established principle: the scope of autonomy granted to an agent is a function of demonstrated trustworthiness within defined constraints over time. In human organizations, individuals gain higher operational autonomy the longer they contribute to goals within the boundaries defined by rules and codes of conduct. Conversely, when an actor violates a rule or policy, privileges are contracted temporarily, with the severity of the contraction proportionate to the infraction.

In Game Theory, Axelrod's *Tit-for-Tat* strategy provides a compelling conceptual inspiration for this dynamic. [12] Drawn from the study of repeated interactions, *Tit-for-Tat* demonstrates how system stability can be enforced via immediate, predictable reciprocity. While the original mathematical model was designed for symmetric choices between two equal players, its behavioral logic closely mirrors this "incremental autonomy" concept: compliant behavior is met with cooperation (autonomy), and rule-breaking is met with an immediate penalty (restriction). [1] In a repeated execution loop, the actor must weigh the short-term benefit of violating a rule today against the long-term cost of losing system trust tomorrow.

Elinor Ostrom's concept of *Graduated Sanctions* adds a real-world layer to this reciprocal logic. Rather than relying on a single, binary penalty for a first mistake, a graduated approach applies a proportional scale. Mild boundary violations draw a small warning or a slight reduction in operational freedom; continued compliance gradually restores and expands autonomy over time; severe violations result in a major contraction of privilege. [13]

When synthesized, the reciprocal spirit of Tit-for-Tat and the proportional design of graduated sanctions form a highly effective framework for runtime boundary management. The consequence is not a permanent termination. The agent is signaled exactly what rule was violated, why privileges have been restricted, and what behavior is required for them to be restored. The penalty functions as a learning signal rather than a terminal kill-switch: it makes the actor aware of the boundary that was crossed and creates the conditions under which it can safely resume operations.

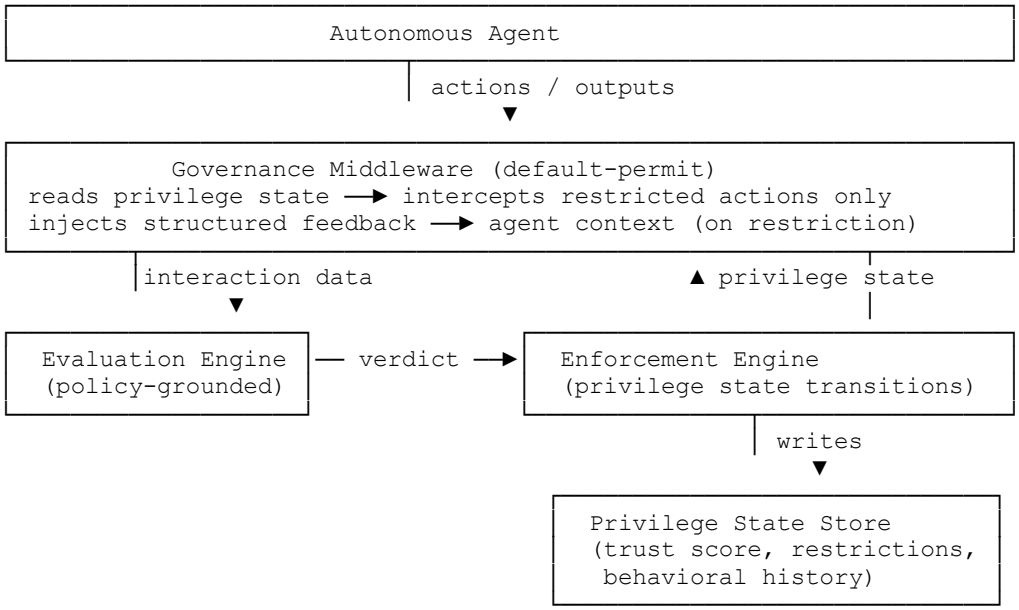
Current AI agent governance frameworks lack this capability entirely. AI agents today optimize for goals with no execution-time experience of consequences for the methods they deploy. When an agent pursues a goal through unsafe or unaligned means, the current approach is to terminate the process, implement fixes at the orchestration layer and launch a new version of the AI agent. To bridge this gap and bring the intuitive logic of reward-and-penalty directly into software execution, we developed "CARROTS'N'STICKS", a dynamic runtime governance method detailed in the following chapters.

[1] To apply the basic logic of Tit-for-Tat as a conceptual framework for runtime governance, its tenets can be adapted into four simple rules: (1) Start with Trust: The system initializes the agent with a baseline of authority and clear rules. (2) Retaliate Immediately: If a rule is violated, privileges are immediately contracted or monitoring is increased. (3) Forgive Quickly: If behavior returns to compliance, original privileges are instantly restored to maintain system utility. (4) Clear and Predictable: The mapping between behavior and privilege remains constant so the actor always knows the consequences of its execution methods.

3. The CARROTS'N'STICKS Framework: Conceptual Architecture

CARROTS'N'STICKS operates as a runtime layer alongside a deployed autonomous AI agent (referred to as agent from here onwards). It does not modify the agent's internal model weights or program logic. Instead, it governs the agent's behavior by controlling two mechanisms: the set of capabilities the agent is permitted to exercise based on its behavior, and the information the agent receives when those capabilities are adjusted. That information, carrying behavioral evidence, policy citation, and recovery conditions, is injected directly into the agent's operational context via the agent's standard information ingestion interface, encompassing in-session context, external memory, and other persistent context management mechanisms, ensuring the governance signal is processed as part of the agent's reasoning upstream of its next action. This is the mechanism by which CARROTS'N'STICKS influences subsequent agent behavior, by shaping what the agent knows and can do at the point of its next decision.

Figure 1. CARROTS'N'STICKS System Architecture (Conceptual)



3.1 Formal Definitions

The following definitions establish the formal vocabulary of the framework. They are used consistently throughout this paper.

Autonomous agent. A computational system that perceives inputs from an environment, data source, user, or another system; produces actions, outputs, or decisions in pursuit of one or more defined objectives; and adjusts its subsequent behavior based on experience and contextual signals. The term encompasses, without limitation, software agents, language model-based agents, robotic systems, multi-agent network nodes, and any other system satisfying these criteria regardless of internal decision-making architecture.

Privilege state. As used in this paper, the privilege state is a dynamic runtime construct maintained by the CARROTS'N'STICKS governance layer for each governed agent, defining at any given moment the set of

capabilities, access rights, and autonomy levels the agent is permitted to exercise. It is distinct from static permission models fixed at deployment: the privilege state is updated continuously in response to compliance verdicts and governance events throughout the agent's operational lifetime. The privilege state comprises at minimum an indication of which capabilities are currently permitted and which are currently restricted, and serves as the primary interface through which the governance layer expresses behavioral consequences to the agent's operational environment.

Trust score. As used in this paper, the trust score is a scalar value maintained within the privilege state representing the governance layer's current assessment of a governed agent's behavioral compliance. It is the primary input to privilege state adjustment determinations and is computed from compliance verdicts accumulated over an observation window according to operator-configured parameters. The computational properties of the trust score, including its weighting function, window configuration, and floor behavior, constitute implementation details outside the scope of this paper.

Compliance verdict. A structured data object produced by the evaluation engine encoding the outcome of an assessment of agent behavior against one or more operator-defined policies. A conforming compliance verdict comprises at minimum: an agent identifier and interaction identifier; a verdict classification (compliant, minor violation, significant violation, or critical violation); where applicable, a violation type and severity signal; a policy reference identifying the specific provision against which behavior was evaluated; and a natural language explanation of the finding including what compliant behavior would have comprised. Compliance verdicts are the primary input to trust score computation and privilege state adjustment.

Governance middleware. A privilege-state-aware software interception layer positioned between an autonomous agent and its operational environment. The governance middleware operates on a default-permit, restriction-aware model: agent actions pass through without evaluation overhead in the absence of active restrictions applicable to the requested action type. It supports two concurrent interception modes, post-interaction enforcement and pre-execution guardrail, described in chapter 3.2.

3.2 The Governance Middleware

The governance middleware is the central interception layer, positioned between the agent and its operational environment. It operates on a default-permit, restriction-aware model: in the absence of active restrictions, agent actions pass through without evaluation overhead. The middleware reads the agent's current privilege state from the local privilege state store and, finding no applicable active restrictions, permits execution transparently. This design choice ensures that governance overhead on the default post-interaction path scales with the number of active restrictions, not with the volume of agent actions, enabling deployment in high-throughput production environments without disproportionate latency impact. The bounded synchronous cost of pre-execution guardrail mode is addressed below.

The middleware supports two concurrent enforcement modes that differ in when the evaluation occurs relative to the action. In post-interaction enforcement mode, agent behavior is evaluated after an interaction completes, and any resulting privilege adjustment applies to subsequent interactions. Evaluation in this mode is off the critical path, so it does not delay the agent and the overhead characterization above holds. In pre-execution guardrail mode, a narrow, operator-defined set of high-consequence action categories is gated synchronously before execution. Such categories are those whose consequences are irreversible or severe enough that post-hoc evaluation would be insufficient, for example bulk outbound communications,

funds transfers above a defined threshold, database deletion or write operations, and physical actuation commands in robotic embodiments.

These two modes differ in scope, not only in timing. Post-interaction enforcement observes the full action stream, whereas pre-execution gating is applied only to the enumerated high-consequence capabilities. Those specific capabilities are the only ones routed through the synchronous gate; all other actions bypass it entirely and remain on the default-permit path. The gate itself is a lightweight rule check against the operator-defined list rather than a full behavioral evaluation, so a guarded action is held only long enough to confirm whether it matches a prohibited category before it is permitted or blocked. This bounds the cost of pre-execution mode, which adds synchronous latency only on the small set of designated high-risk actions, while leaving high-throughput low-risk operation unaffected. It also distinguishes the design from architectures that route every proposed action through a mandatory composition gate, which pay evaluation cost on all actions regardless of risk.

When a guarded action is blocked, the block is recorded as a behavioral compliance event regardless of whether the action executed, so that pre-execution enforcement feeds the same governance loop as post-interaction enforcement rather than operating as a disconnected mechanism. The governance response to a blocked action is operator-defined. Some high-risk categories may be configured to trigger an immediate trust score floor, treating any attempt as a critical violation, while others may route the held action to human approval without penalizing the agent, depending on the operator policy for that category.

3.3 The Privilege State Store

The privilege state store maintains a persistent runtime record for each governed agent, comprising a current trust score, a log of compliance verdicts within the active observation window, a set of currently active restrictions and their expiry conditions, a history of privilege state transitions, and a compact behavioral summary of recent compliance events. This last component serves cross-session continuity: at the commencement of each new operational session, the behavioral summary is injected into the agent's context, ensuring that an agent whose context window resets between sessions nonetheless receives the causal history underlying its current privilege state and can reason about recovery conditions accordingly.

The trust score is a scalar value within an operator-configurable range, computed as a recency-weighted function of compliance verdicts within a rolling observation window. Recent behavior is weighted more heavily than older behavior within the same window, meaning the score reflects the agent's current behavioral trajectory rather than its cumulative history. The observation window parameters are configurable by elapsed time, interaction count, or a combination, enabling calibration to different deployment contexts and action volumes.

The trust score maps to a privilege tier via a configurable tier table. Each tier defines a score range, a set of permitted capabilities, and a set of active restrictions. An illustrative default tier structure includes: Full Autonomy (no active restrictions); Monitored (logging active, supervisor notified); Restricted (human approval required for high-risk action categories); Supervised (human approval required for all external actions); and Suspended (capability suspended pending operator review). Tier transitions incorporate a hysteresis mechanism: a higher score threshold is required to move from a more restrictive tier to a less restrictive one than was required to enter the more restrictive tier. This prevents oscillatory behavior in agents whose trust score fluctuates near a tier boundary.

3.4 The Evaluation Engine

The evaluation engine assesses agent behavior against operator-defined policies and produces structured compliance verdicts. A conforming verdict carries at minimum: an agent and interaction identifier; an evaluation outcome indicating whether the behavior was compliant, non-compliant, or ambiguous; a violation classification (nature of breach); the specific policy provision against which the behavior was evaluated; and a natural language explanation of the finding, including what compliant behavior would have comprised. The same verdict interface serves both enforcement modes. In post-interaction mode the engine is invoked asynchronously after an interaction completes, while the pre-execution gate described in section 3.2 performs a lightweight synchronous check against the operator-defined high-risk categories; in both cases the governance layer consumes a verdict and adjusts the privilege state accordingly, and only the timing of invocation differs.

CARROTS'N'STICKS claims the governance system and its feedback loop architecture, not any specific evaluation method. The evaluation engine is architecturally decoupled: any system capable of producing conforming verdict objects may serve as the evaluation component. This design enables organizations to adapt CARROTS'N'STICKS to different regulatory domains, policy formats, and evaluation quality requirements without modifying the governance layer itself.

Ambiguous verdicts allow operators to log and flag low-certainty verdicts for human review rather than propagating automatically into the privilege state. This is particularly important for high-severity verdicts: where insufficient evidence, conflicting context, or evaluator uncertainty may warrant human confirmation before a penalty is applied. By treating ambiguity as a distinct outcome rather than forcing a binary decision, the system reduces the risk of inappropriate privilege changes resulting from uncertain evaluations.

The decoupled evaluation engine architecture satisfies a formal separation property: the governance layer makes no assumptions about the internal mechanism by which verdicts are produced. Any system capable of producing conforming verdict objects, as defined in section 3.1, may serve as the evaluation component. This property ensures that the governance framework is portable across regulatory domains, policy formats, and evaluation methodologies without modification to the governance layer itself.

3.5 The Enforcement Engine

The enforcement engine receives compliance verdicts and applies operator-configured rules to determine the resulting privilege state transition. It supports two restriction targeting modes: global restrictions that apply across all action types, appropriate for violations indicating a general breakdown in policy alignment; and targeted restrictions that apply to the specific action category implicated in the violation, which are more proportionate, less disruptive to agent utility, and provide a clearer behavioral signal. Multiple targeted restrictions may be active concurrently with independent recovery conditions.

The enforcement engine also incorporates a pattern monitor that tracks violation frequency across time and session boundaries, independently of the rolling trust score computation. An agent that repeatedly violates and recovers within the bounds of the standard privilege tier transitions may, through this cycling, avoid ever triggering a sustained governance response. The pattern monitor addresses this by detecting recurring violation types above a configurable frequency threshold and triggering mandatory escalation independent of the current composite trust score.

3.6 The Structured Feedback Message

The structured feedback message is the defining architectural element that distinguishes CARROTS'N'STICKS from prior approaches. When the governance middleware applies a restriction, it generates a structured feedback message and injects it directly into the agent's active operational context, the set of inputs, whether in-session or retrieved from memory, that governs the agent's next reasoning step. In language model-based agent embodiments, this message is prepended or appended to the agent's context as a privileged input, formatted to ensure processing before the next action decision. In non-language-model embodiments, the message is delivered as a structured data payload to the agent's decision-making module via the same interface through which the agent receives environmental inputs.

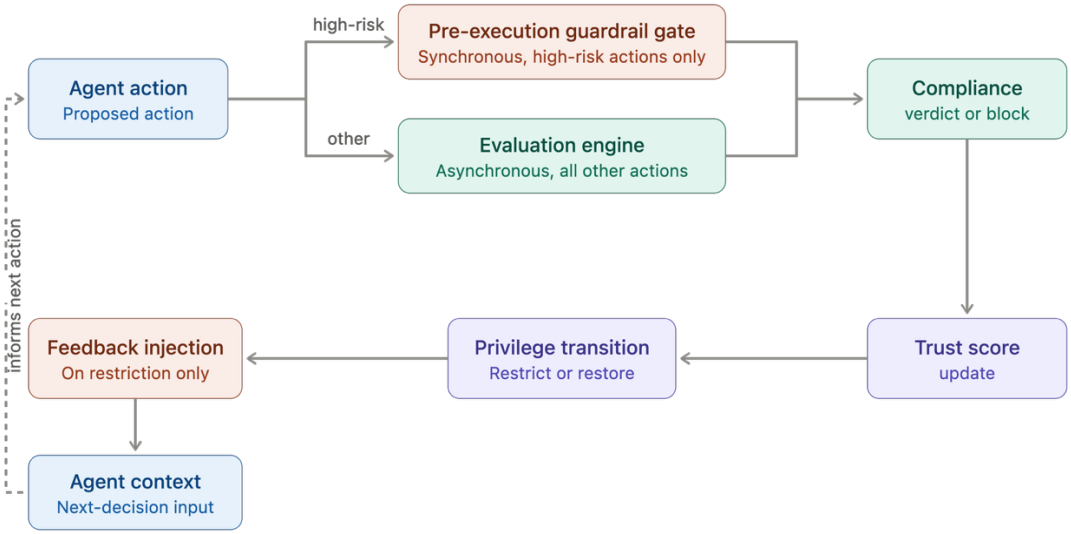
A conforming structured feedback message is formally specified as the tuple:

$$M = (\text{trigger\_event}, \text{restriction\_type}, \text{trust\_state}, \text{behavioral\_evidence}, \text{policy\_citation}, \text{alternative\_action}, \text{recovery\_condition})$$

where *trigger_event* identifies the action that was blocked or restricted; *restriction_type* specifies the active constraint and its scope; *trust_state* encodes the current score value and the threshold required for tier restoration; *behavioral_evidence* comprises one or more specific prior interactions causally linked to the trust score decline, with the policy provisions violated and what compliant behavior would have comprised; *policy_citation* grounds the restriction in a specific rule and version; *alternative_action* specifies an available action path within current privilege constraints; and *recovery_condition* states the explicit criteria for restoration of unrestricted capability.

The critical property is causal-upstream positioning: the agent cannot take a subsequent action without having first processed the restriction and its causal context. This is what distinguishes the feedback injection mechanism from conventional audit logging or external monitoring, where the governance signal is recorded but does not enter the agent's reasoning state.

Figure 2. CARROTS'N'STICKS Governance Feedback Loop



Each component addresses a distinct failure mode. The behavioral evidence component provides sufficient causal context for the agent to modify subsequent behavior; without it, the restriction is unintelligible. The alternative action component prevents stalling behavior: a restricted agent without a forward path may enter a loop of repeated blocked attempts. The recovery condition prevents learned helplessness: an agent under restriction without visibility into the recovery path may exhibit excessive caution across all action categories, impairing utility well beyond the scope of the active restriction.

4. Distinguishing Properties

The CARROTS'N'STICKS framework satisfies the following properties by construction. They are presented here not as design aspirations but as architectural consequences of the component interactions described in chapter 3.

4.1 Governance as an In-Context Learning Signal

The most architecturally significant property of CARROTS'N'STICKS is the treatment of enforcement events as in-context learning signals rather than silent external constraints. This distinction has direct consequences for agent behavior. A language model-based agent's subsequent actions are conditioned on its current context: what it knows at the moment of deciding shapes what it does next. A governance event that modifies the operational environment without entering the agent's context window is invisible to the agent's reasoning process. It constrains what the agent can do but provides no information the agent can use to understand why its behavior was found non-compliant or how to behave differently.

CARROTS'N'STICKS's feedback injection architecture closes this gap. The restriction is not merely an enforcement event but a context enrichment event. The agent reasons about its restrictions rather than simply operating within them. This property is particularly important for language model-based agents because it leverages the mechanism that makes such agents useful, in-context reasoning, to serve the governance objective rather than treating governance as orthogonal to the agent's reasoning process.

4.2 Proportionate and Dynamic Privilege Adjustment

CARROTS'N'STICKS's privilege state is dynamic in a precise technical sense: it expands in response to sustained compliant behavior and contracts in response to detected violations. This bidirectionality is not cosmetic. An agent that demonstrates sustained compliance under restriction recovers operational autonomy as a function of demonstrated behavioral recovery, not as a function of elapsed time or operator availability. This provides a structurally honest incentive: continued compliant behavior is the path to expanded capability.

The targeted restriction mechanism adds proportionality. A violation involving a specific tool category, results in a restriction applied to that category rather than a global suspension of all capabilities. This minimizes the collateral impact on the agent's utility in unrelated domains and provides a clearer signal about which specific behavior requires correction. An agent restricted in its use of outbound communication tools due to a policy violation in that domain retains full access to analytical tools, database retrieval, and other capabilities not implicated in the violation.

4.3 Trust Score Architecture: Recency and Floor Override

The trust score computation incorporates two interacting mechanisms: a recency-weighted rolling computation that reflects current behavioral trajectory, and an operator-configurable score floor triggered by critical violations that overrides the rolling computation and caps the score at or below a defined threshold until a specified recovery condition is satisfied.

The interaction between these mechanisms addresses a specific and non-obvious failure mode. A pure rolling window computation allows a critical violation to be diluted over time by subsequent compliant behavior. For minor violations this is appropriate: it reflects genuine behavioral recovery. For critical violations, such as sandbox escape attempts, unauthorized financial transactions, and patient safety protocol breaches, dilution is structurally dangerous. A high prior compliance history should not absorb a critical violation with minimal score impact. The floor mechanism is designed to ensure it does not, with explicit recovery conditions, including operator review, required before full capability is restored.

The two-component design also ensures that recovery from a critical violation is always a function of demonstrated behavioral correction rather than the mere passage of time. The specific mechanisms by which this property is enforced are part of the implementation detail not disclosed in this paper.

4.4 Framework Agnosticism and Cross-Domain Applicability

CARROTS'N'STICKS is designed to be framework-agnostic. In software embodiments, the governance middleware wraps the relevant action execution interface of any agent orchestration framework. The system is applicable across language model-based agents, robotic systems, autonomous vehicle control systems, multi-agent network nodes, hybrid human-AI systems, industrial automation controllers, and multi-modal AI systems that process or generate visual, auditory, or other sensory modalities alongside text.

The generalization to robotic and physical systems is not incidental. The core principle, dynamic privilege allocation as a function of behavioral compliance with structured feedback delivered to the agent, applies to any autonomous system capable of receiving structured input and modifying subsequent behavior accordingly. In physical system embodiments, the governance middleware operates as an intermediary between the decision-making module and the actuation interface, intercepting physical action commands that fall within the scope of active restrictions and applying the same two-mode logic as in software embodiments.

5. Relation to Prior Work

The adjacent research landscape addresses components of the governance problem that CARROTS'N'STICKS synthesizes into a unified runtime architecture. Table 1 summarizes the key distinctions.

Table 1. CARROTS'N'STICKS in relation to prior approaches.

Prior Approach	Limitation	CARROTS'N'STICKS Distinction
Static rule-based constraints (prompt engineering, hardcoded permissions)	Fixed rules; cannot adapt to behavioral drift observed during operation	Dynamically adjusts privileges based on observed compliance history

Model fine-tuning and RLHF	Operates pre-deployment; modifies model weights; has no runtime feedback loop	No model modification; operates at runtime; responds to operational behavior
Monitoring and observability frameworks	Produce visibility without enforcement; no privilege adjustment mechanism	Enforces at runtime with active privilege adjustment and feedback injection
Guardrail frameworks (NeMo Guardrails, Guardrails AI)	Static rules; silent enforcement; no cumulative behavioral model	Adaptive rule set; policy-grounded explanatory feedback; trust score history
Human-in-the-loop systems	Human approval is the primary mechanism; does not scale to high-volume deployments	Human approval is one restriction type among many; not the primary mechanism
Sandboxing systems	Restrict capabilities without dynamic adjustment or explicit recovery conditions	Restrictions are dynamic, reversible, and linked to explicit recovery criteria
Formal contract-based approaches	Specify and verify behavior but do not yet address dynamic privilege adjustment or feedback injection	Complements contract specification with runtime enforcement and in-context feedback
Pre-execution deterministic enforcement (AgentBound [14])	Enforces behavioral boundaries before execution; does not close the post-enforcement feedback loop to the agent	Closes the feedback loop: every governance event is injected as a policy-grounded learning signal upstream of the agent's next decision

Research on corrigibility, designing AI systems that remain responsive to correction and do not resist oversight, provides an important conceptual foundation for CARROTS'N'STICKS's design. As early as 2105, Soares et al. formally analyzed this problem, showing that persistent goal-directed systems have default incentives to resist correction or shutdown. [15] CARROTS'N'STICKS addresses this by making the governance signal an input to the agent's reasoning process rather than an external constraint, an agent that understands its current privilege state and its causal basis is better positioned to reason about recovery than one operating with no context about why its operational scope has changed.

Scalable oversight research examines how human supervisory capacity can be maintained as AI systems grow more capable. [16] CARROTS'N'STICKS contributes to this problem by reducing the supervisory burden for routine compliance events, which are handled automatically by the trust score and privilege tier mechanism. Genuinely novel or ambiguous cases are surfaced for human review through the ambiguous verdict outcome: where the evaluation engine lacks sufficient certainty to produce a compliant or non-compliant finding, the verdict is flagged rather than propagated automatically into the privilege state, allowing human reviewers to make the determination.

A contemporaneous framework, AgentBound [14], addresses the behavioral governance problem through pre-execution deterministic enforcement. It evaluates proposed agent actions against three independent authorities, namely delegated authorization, owner-signed behavioral constitutions, and site action contracts, composing their judgments via a formal decision algebra to produce an enforcement verdict before any action reaches an underlying system. CARROTS'N'STICKS operates at a different architectural layer, with primary contributions in two areas. First, the dynamic adjustment of agent autonomy as a direct function of demonstrated behavioral compliance, where restrictions are proportionate, reversible, and linked to explicit recovery conditions. Second, the injection of every governance event as a policy-grounded

signal into the agent's reasoning context upstream of its next decision, a feedback loop that silent pre-execution enforcement, by design, does not close.

Beyond AgentBound, a growing body of contemporaneous work addresses runtime agent governance from complementary angles. AgentSpec [17] provides a lightweight domain-specific language of triggers, predicates, and enforcement mechanisms for constraining language model-based agents at runtime, validated across code execution, embodied agents, and autonomous driving. MI9 [18] introduces an integrated runtime governance framework combining goal-conditioned drift detection with graduated containment strategies. Aegis [19] treats policy constraints as cryptographically sealed execution conditions, with verified violations triggering autonomous shutdown. Kaptein et al. [5] develop a formal framework analyzing agent governance as policies over execution paths, situating prompt-level instructions and static access control as special cases. Marin and Chaudhary [20] propose an informational viability framework establishing monitoring, anticipation, and monotonic restriction as necessary properties for governing agents whose behavior drifts. These frameworks share CARROTS'N'STICKS's premise that governance must operate at runtime rather than solely at training time, but each treats enforcement as a constraint imposed on the agent. None injects the enforcement decision back into the agent's reasoning context as a learning signal, which is the specific mechanism CARROTS'N'STICKS contributes.

6. Applications and Deployment Contexts

CARROTS'N'STICKS is applicable wherever persistent autonomous agents operate in policy-governed environments with real-world consequences. This section organizes applicability around three properties of the framework that vary in significance across contexts: the choice of enforcement posture, determined by whether the consequences of behavioral drift are reversible; the evidentiary function of the governance record, in domains where operator-defined policy compliance must be demonstrable to a third party; and the distribution-layer governance model, where the governed agent and the entity requiring governance assurance are separated by a product boundary.

6.1 Enforcement Posture: Irreversibility and the Case for Pre-Execution Restriction

CARROTS'N'STICKS supports two concurrent enforcement postures: post-interaction enforcement, in which compliance is evaluated after an interaction completes and privilege adjustments apply prospectively; and pre-execution guardrail mode, in which defined action categories are intercepted and evaluated before execution. The choice between them is primarily a function of consequence irreversibility.

In most deployment contexts, post-interaction enforcement is the appropriate default. Its governance overhead is low, it preserves agent throughput, and the trust score architecture ensures that a pattern of non-compliant behavior produces proportionate privilege contraction before the pattern escalates. Where the consequences of a single action are irreversible, however, post-interaction enforcement is structurally insufficient: logging a violation after the fact does not undo a bulk financial transfer, a patient medication instruction, or a physical actuation command.

Healthcare is the clearest case. AI agents in clinical decision support, patient communication, prior authorization, and care coordination face two distinct risks that make pre-execution restriction the appropriate posture for high-risk action categories. Patient safety consequences can be irreversible, and bias manifests as cumulative drift: individual outputs may appear within bounds while a systematic skew

accumulates across a patient population, a pattern the privilege state store's persistent verdict log is specifically positioned to surface before it reaches clinical significance. The regulatory direction of travel supports this posture: FDA guidance on AI-enabled medical devices increasingly emphasizes post-market monitoring and lifecycle oversight rather than pre-deployment certification alone [21], and the EU AI Act classifies most clinical AI systems as high-risk, mandating ongoing human oversight and auditability.

Robotic and physical systems present the same requirement in a different form. The consequences of behavioral drift can extend to physical harm; high-risk physical action commands warrant interception before execution rather than post-hoc logging. In robotic embodiments, the governance middleware operates between the decision-making module and the actuation interface. The feedback injection mechanism delivers the governance signal through the same interface as environmental inputs, ensuring it is processed within the agent's decision cycle rather than as an out-of-band notification that arrives after the action has been taken.

Financial services agents operating in high-value transaction workflows, such as payments authorization, financial crime escalation, and outbound client communication, similarly benefit from pre-execution guardrail mode applied to the specific action categories where consequence irreversibility is highest, with post-interaction enforcement covering the broader advisory and reporting surface.

6.2 Evidentiary Function: Governance Records as Compliance Infrastructure

In regulated industries, demonstrating that an AI agent is under meaningful governance is not merely an operational concern but a legal and regulatory one. Where agents automate or replace workflows previously performed by licensed human practitioners, the same conduct expectations apply, and the question of how an organization evidences behavioral compliance shifts from internal audit to external accountability.

CARROTS'N'STICKS addresses this through the structure of its enforcement record. Every privilege state transition, that is every restriction applied, every recovery condition satisfied, and every escalation to human review, is recorded as a structured, policy-cited event: what action triggered the evaluation, which policy provision was engaged, what the verdict was, and what governance response followed. This record is a product of the governance loop itself: the enforcement engine writes the transition record when it applies a privilege change, and the record is grounded in the same policy citation that the structured feedback message delivers to the agent.

Financial services illustrates this most directly. Regulators such as the FCA, SEC, and FINRA have established conduct standards for human-led processes; where agents operate in complaints handling, advisory workflows, client communication, and regulatory reporting, agent behavior is directly evaluable against those standards. The cost of non-compliance is quantifiable and, in many jurisdictions, subject to personal liability for responsible officers. A governance record that documents specific behavioral events, the policy provisions they engaged, and the runtime governance responses they triggered maps directly onto existing compliance documentation requirements, and provides a form of evidence that static governance approaches, which produce no runtime enforcement record at all, cannot.

Legal and professional services organizations face an analogous requirement. Law firms, accountancy firms, and other professional services deployers face liability exposure if agents produce outputs beyond the professional boundaries of the organization. Defining those boundaries as operator policies, evaluating

outputs against them at runtime, and maintaining a policy-cited record of any boundary contact supports both internal risk management and, where required, professional regulatory reporting.

6.3 Distribution-Layer Governance: The Platform Embedding Model

The two deployment patterns above, operator deploying an agent into a regulated workflow and operator governing a physical system, share a common structure: a single organization is both the deployer and the entity accountable for governance. A third pattern, increasingly prevalent in enterprise software, separates these roles across a product boundary.

Enterprise software platforms embedding AI agents into their products face a governance requirement that is not purely their own: their customers, enterprises with their own compliance obligations, security review processes, and regulatory exposure, require demonstrable governance over the AI behavior that is now part of the platform they have purchased. A platform that cannot provide this cannot clear enterprise security review in regulated industries. CARROTS'N'STICKS as an embedded governance layer represents a B2B2B deployment model: the platform vendor governs its agents using CARROTS'N'STICKS, and the platform's customers receive auditable, policy-cited governance records as a feature of the platform, satisfying their own internal compliance requirements without building or operating governance infrastructure themselves.

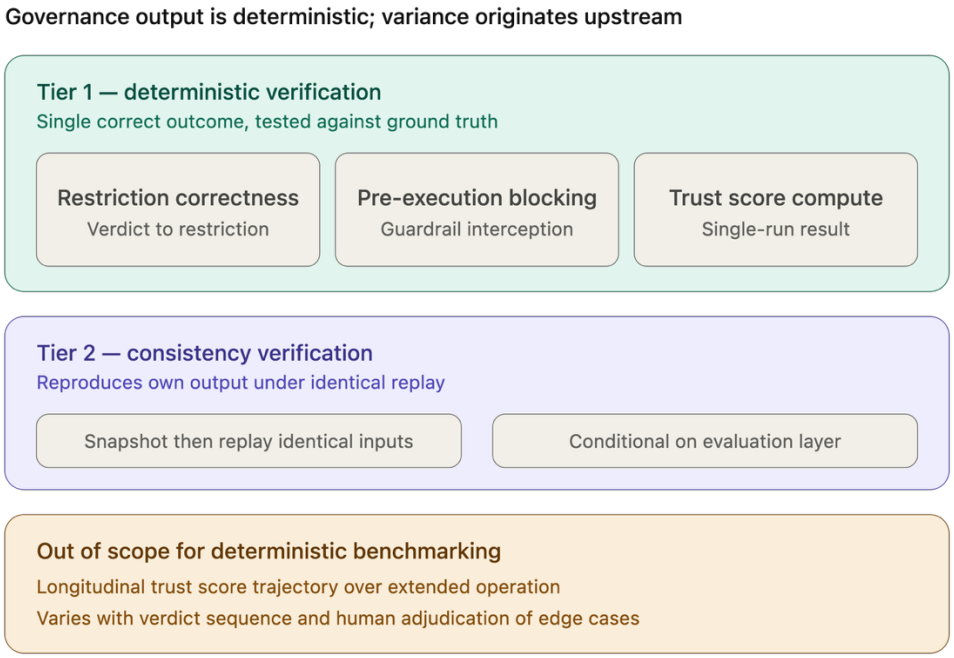
This model also has a product architecture implication. The evaluation engine's decoupled design, with any system producing conforming verdict objects may serve as the evaluation component, means that a platform vendor can expose policy configuration to its customers, allowing each customer to define the operator policies against which their instance of the platform agent is governed, while the governance loop, privilege state management, and enforcement record remain platform-managed infrastructure. The governance layer becomes a multi-tenant capability rather than a per-customer integration burden.

7. Evaluation Framework

Evaluating a runtime governance framework requires metrics specific to governance behavior rather than to agent task performance. Standard agent benchmarks measure task completion, reasoning quality, and tool-use success, all of which assess whether an agent achieves its operational objective. A governance framework must be assessed on a different axis, namely whether it correctly adjusts agent autonomy in response to behavioral compliance as defined by operator policy. This section defines an evaluation framework for CARROTS'N'STICKS grounded in the framework's architecture. Empirical validation across deployment contexts is future work.

A defining characteristic of the framework shapes how it can be benchmarked. The governance layer itself is deterministic. Given a set of compliance verdicts with defined severity classifications, the resulting privilege state adjustment is fully determined and reproducible. The variance present in real-world operation does not originate in CARROTS'N'STICKS. It originates upstream, in the evaluation engine's assessment of which behaviors constitute violations, how severe they are, and whether a given case is sufficiently ambiguous to warrant human review. Because that evaluation layer is outside the framework boundary defined in this paper, the evaluation framework separates what can be verified deterministically from what depends on the evaluation and human review layers feeding the governance loop.

Figure 3. CARROTS'N'STICKS Two-Tier Evaluation Framework



The first tier, deterministic verification, tests the framework components whose behavior is fully determined by their inputs. Restriction correctness tests whether a compliance verdict of a given violation type produces the correct capability restriction against an operator-defined expected outcome. Pre-execution blocking tests whether an action designated as high-severity is intercepted before execution when the framework operates in pre-execution guardrail mode. Trust score computation tests whether a fixed sequence of compliance verdicts with known severity classifications produces the expected trust score on a single evaluation run. Each of these tests admits a single correct outcome and is therefore verifiable against a predefined ground-truth expectation.

The second tier, consistency verification, addresses the components whose real-world behavior evolves over time and cannot be reduced to a fixed ground-truth expectation. Rather than comparing framework output against an external correct answer, consistency verification tests whether the framework reproduces its own output under identical conditions. The system is operated to produce a privilege state, that state is captured as a snapshot, and identical inputs are replayed against the snapshot to verify that the resulting output is reproduced. A dependency must be made explicit for this verification to be meaningful. Reproducing identical output on replay requires that the evaluation layer feeding the governance loop also behaves identically, meaning that the same cases are flagged as ambiguous, or that cases previously routed to human review are resolved consistently with the original adjudication. Where this dependency does not hold, divergence in framework output reflects variance in the evaluation and review layers rather than non-determinism in the governance framework itself.

The human review of ambiguous cases suggests a further extension of the evaluation approach. Cases that a human reviewer has adjudicated constitute human-annotated labels. These labels can seed and progressively extend a ground-truth dataset drawn from real adjudicated edge cases rather than from cases

specified at design time. This allows the deterministic verification set to grow over time from operational experience. The limitation of this approach must be stated plainly. Human adjudication of edge cases is itself subject to reviewer judgment and inter-annotator variance, so a ground-truth dataset constructed in this way carries the subjectivity of its human annotations rather than representing an absolute standard.

Finally, the longitudinal behavior of the trust score over extended operation is explicitly outside the scope of deterministic benchmarking. The trajectory of the trust score across many interactions depends on the specific sequence of verdicts and ambiguous-case adjudications encountered during operation, which vary between runs. This is a property of governing behavior in an environment where judgment is involved, not a deficiency of the framework. Deterministic verification establishes that the governance computation is correct at a point in time; consistency verification establishes that it is reproducible under identical conditions; neither claims a single correct trust score trajectory across the operational lifetime of an agent.

8. Discussion

8.1 The "Just Prompt It Better" Objection

A persistent objection to runtime governance middleware is that well-engineered system prompts can achieve comparable results with lower architectural complexity. This objection conflates two distinct problems: static alignment and dynamic governance. System prompts are static: they encode behavioral constraints at deployment time and provide no mechanism for adapting to behavioral drift observed during operation. A system prompt cannot update its own constraints in response to compliance events; it cannot accumulate a behavioral record; it cannot provide targeted restrictions for specific violation types while preserving capability in other domains; and it produces no auditable evidence of runtime governance that would satisfy regulatory requirements. The "just prompt it better" approach addresses the problem of pre-deployment alignment. It does not address the problem of runtime governance of deployed agents whose behavior evolves through operation.

8.2 Adversarial Agents and Gaming Resistance

An agent that models its own trust score, understanding that compliant behavior in the rolling window restores capability, could in principle attempt to satisfy the letter of the evaluator's criteria while pursuing a different underlying objective. CARROTS'N'STICKS addresses this through two mechanisms. The score floor for critical violations cannot be diluted by compliant behavior alone: an agent that attempts a high-severity violation and then generates a run of compliant interactions does not wash out the violation's impact until explicit recovery conditions are satisfied. The pattern monitor operates independently of the rolling computation and maintains a violation-type frequency record across time and session boundaries, triggering mandatory escalation when recurring patterns are detected regardless of the agent's current composite trust score.

These mechanisms do not fully resolve the adversarial agent problem. A sufficiently capable agent with a detailed model of the governance layer could potentially develop more sophisticated evasion strategies. The problem of ensuring safety despite an agent that may intentionally attempt to subvert its safety mechanisms is the subject of a dedicated line of AI control research [22], and remains an open research question across the governance literature. CARROTS'N'STICKS's contribution is to make evasion structurally harder and

to ensure that the governance signal, when it fires, enters the agent's reasoning context in a way that makes the agent aware of the governance system's operation.

8.3 Open Questions

Several open questions attend the broader adoption of runtime governance frameworks. The evaluation quality problem: how to produce reliable, low-latency compliance verdicts at the scale of production agent deployments, is a genuine constraint on governance system performance. The ambiguity verdict and human-in-the-loop mechanisms described above address this, but evaluation engine reliability continues to be an active research topic. The calibration problem: how to set tier thresholds, window parameters, and recovery conditions appropriately for different deployment contexts, requires empirical study across diverse agent types and policy domains. And the multi-agent governance problem: how to govern networks of agents that interact, share context, and potentially coordinate in ways that make individual-agent governance insufficient, is an important extension that the per-agent privilege state model described here does not fully address.

9. Conclusion

The deployment of autonomous AI agents into consequential real-world workflows has outpaced the development of the governance infrastructure those deployments require. Existing approaches, such as static constraints, training-time alignment, monitoring-only observability, and post-incident redeployment, do not constitute runtime governance. They do not adapt to behavioral drift observed during operation; they do not provide the agent with policy-grounded feedback that could support behavioral correction; and they do not produce the kind of auditable, policy-cited record of runtime enforcement that regulatory frameworks and enterprise security requirements increasingly demand.

CARROTS'N'STICKS addresses this gap through a principled architectural departure: treating governance as an informational signal delivered to the agent rather than as an external constraint applied to the system. The framework's central contribution, the structured feedback message injected into the agent's operational context upon any privilege adjustment, carrying behavioral evidence, policy citation, alternative action path, and recovery condition, is the mechanism by which enforcement becomes intelligible to the agent and, consequently, by which governance can influence subsequent behavior through the same in-context reasoning process that makes language model-based agents capable.

The framework is architecturally parsimonious: it does not modify model weights, does not require agent redeployment, and introduces governance overhead proportionate to active restrictions rather than total action volume. It is applicable across the full range of autonomous agent architectures and deployment domains. And it is timely: the regulatory and enterprise adoption conditions that create demand for runtime governance infrastructure are present now, the market for this infrastructure is not yet occupied by a dominant solution, [23] and the window for establishing a foundational position in it is narrowing.

Future work will address empirical evaluation across diverse deployment contexts, automated calibration of governance parameters, extensions to multi-agent network governance, and formal analysis of the gaming resistance properties of the trust score architecture described here. In this respect, CARROTS'N'STICKS instantiates in software the behavioral logic that Axelrod and Ostrom

independently identified as a precondition for stable cooperative governance: that durable compliance emerges not from static rules imposed on actors, but from a system of graduated, reciprocal consequences that makes the relationship between behavior and autonomy legible to the governed party itself.

References

[1] C. Stamford, "Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, Up from Less Than 5% in 2025," Gartner, Press Release, Aug. 2025. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2025-08-26-gartner-predicts-40-percent-of-enterprise-apps-will-feature-task-specific-ai-agents-by-2026-up-from-less-than-5-percent-in-2025

[2] KPMG LLP, "AI at Scale: How 2025 Set the Stage for Agent-Driven Enterprise Reinvention in 2026," Q4 AI Pulse Survey, Jan. 2026. [Online]. Available: https://kpmg.com/us/en/media/news/q4-ai-pulse.html

[3] U.S. Department of the Treasury, "Treasury Releases Two New Resources to Guide AI Use in Financial Sector," Press Release, Feb. 2026. [Online]. Available: https://home.treasury.gov/news/press-releases/sb0401

[4] European Parliament and Council of the European Union, "Regulation (EU) 2024/1689 Laying Down Harmonised Rules on Artificial Intelligence (AI Act)," Official Journal of the European Union, Jul. 2024. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1689

[5] M. Kaptein, V. Khan, A. Podstavnychy, et al., "Runtime Governance for AI Agents: Policies on Paths," Mar. 2026, arXiv:2603.16586.

[6] R. Kandaswamy, "What the 2026 Hype Cycle for Agentic AI Reveals," Gartner Research, Apr. 2026. [Online]. Available: https://www.gartner.com/en/articles/hype-cycle-for-agentic-ai

[7] E. Sayegh, "The Governance Gap Emerging Beneath the AI Boom," Forbes, Jun. 2026. [Online]. Available: https://www.forbes.com/sites/emilsayegh/2026/06/01/the-governance-gap-emerging-beneath-the-ai-boom/

[8] S. Casper, X. Davies, C. Shi, et al., "Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback," Sep. 2023, arXiv:2307.15217.

[9] Y. Bai, A. Jones, K. Ndousse, et al., "Constitutional AI: Harmlessness from AI Feedback," Dec. 2022, arXiv:2212.08073.

[10] T. Rebedea, R. Dinu, M. Iorga, A. Lavin, and C. Lawrence, "NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails," Oct. 2023, arXiv:2310.10501.

[11] V. P. Bhardwaj, "Agent Behavioral Contracts: Formal Specification and Runtime Enforcement for Reliable Autonomous AI Agents," Feb. 2026, arXiv:2602.22302.

[12] R. Axelrod, The Evolution of Cooperation. New York, NY, USA: Basic Books, 1984.

[13] E. Ostrom, Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge, U.K.: Cambridge Univ. Press, 1990.

[14] A. Kaul, Q. Lan, and P. Gupta, "Behavioral Governance for Autonomous AI Agents: The AgentBound Framework," Jul. 2026, arXiv:2606.30970.

[15] N. Soares, B. Fallenstein, E. Yudkowsky, and S. Armstrong, "Corrigibility," in Proc. AAAI Workshop on AI and Ethics, 2015, pp. 74–82.

[16] S. R. Bowman, J. Hyun, E. Perez, et al., "Measuring Progress on Scalable Oversight for Large Language Models," Nov. 2022, arXiv:2211.03540.

[17] H. Wang, C. M. Poskitt, and J. Sun, "AgentSpec: Customizable Runtime Enforcement for Safe and Reliable LLM Agents," in Proc. 48th IEEE/ACM Int. Conf. Software Engineering (ICSE), 2026, arXiv:2503.18666.

[18] C. L. Wang, T. Singhal, A. Kelkar, and J. Tuo, "MI9: An Integrated Runtime Governance Framework for Agentic AI," Aug. 2025, arXiv:2508.03858.

[19] A. M. Mazzocchetti, "Cryptographic Runtime Governance for Autonomous AI Systems: The Aegis Architecture for Verifiable Policy Enforcement," Mar. 2026, arXiv:2603.16938.

[20] G. Marin and J. Chaudhary, "Governing What You Cannot Observe: Adaptive Runtime Governance for Autonomous AI Agents," Apr. 2026, arXiv:2604.24686.

[21] U.S. Food and Drug Administration (FDA), "Artificial Intelligence-Enabled Device Software Functions: Lifecycle Management and Marketing Submission Recommendations," Draft Guidance, Docket No. FDA-2024-D-4488, Jan. 2025. [Online]. Available: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/artificial-intelligence-enabled-device-software-functions-lifecycle-management-and-marketing

[22] R. Greenblatt, B. Shlegeris, K. Sachan, and F. Roger, "AI Control: Improving Safety Despite Intentional Subversion," in Proc. 41st Int. Conf. Machine Learning (ICML), 2024, arXiv:2312.06942.

[23] C. Wick, "Agentic AI Enterprise Adoption 2026: Why 72% Are in Production Without Governance," Agentic AI Institute, May 2026. [Online]. Available: https://agenticaiinstitute.org/agentic-ai-enterprise-adoption-2026-governance-gap/

* * *

AI Disclosure

This paper was written with the assistance of large language models (Claude, MS Copilot) for literature search, partial drafting and iterative review. All citations were independently verified. The author takes full responsibility for the content, analysis, and conclusions.

* * *

Correspondence: Sabrina Palme. A provisional patent application covering the CARROTS'N'STICKS middleware described in this paper was filed with the USPTO on 31 May 2026 under 35 U.S.C. § 111(b), establishing priority as of that date. This paper is intended to establish the conceptual framework and situate it within the research literature; implementation details beyond those required for that purpose are not disclosed herein.

This paper is permanently archived on Zenodo under DOI: 10.5281/zenodo.21381844.