Online ISSN: 3104-9338
Print ISSN: 3104-932X

Recursive Adversarial Engine: A Stability-Oriented Architecture for Reliable Large Language Model Multi-Agent Systems

Jianhua Fang

Shihao Nine Laboratory

Received: Oct 15, 2026

Revised: Oct 21, 2026

Accepted: Oct 26, 2026

Published online:October 30, 2026

To appear in: International Journal of Advanced AI Applications, Vol. 2, No. 11 (November 2026)

∗ Corresponding Author:

JianhuaFang

(shardylab@sina.com)

Abstract. Large language model (LLM) based multi-agent systems are rapidly migrating from laboratory prototypes to production deployments in finance, scientific research, industrial digital twins, and urban scheduling. As deployment scales, however, a family of systemic pathologies persists: decision oscillation driven by zero-sum agent interactions, cascade propagation of hallucinated content, long-horizon behavioral drift, value misalignment, and unresolved inter-agent conflict. Existing orchestration paradigms, which rely on instruction dispatch, greedy optimization, or static rules, address only shallow coordination and provide no principled mechanism for convergence. This paper proposes the Recursive Adversarial Engine (RAE), a runtime architecture for multi-agent systems organized around three propositions: bounded adversarial interaction supplies the error-detection signal, recursive correction supplies the error-elimination mechanism, and dynamic checks and balances supply the stability guarantee. We specify a four-layer reference architecture comprising adversarial verification, recursive correction, checks-and-balances scheduling, and steady-state output, and we characterize its convergence behavior through a Lyapunov-style stability argument. A qualitative comparison with self-play, multi-agent debate, alignment, and consensus frameworks delineates the paradigm difference. Engineering implications for high-stakes domains, long-horizon robustness, auditability, and middleware integration are analyzed, and open challenges are identified.

Keywords: *Convergence; Hallucination; Autonomy; Checks and balances; Robustness; Interpretability*

1. Introduction

The past three years have witnessed a decisive architectural shift in applied artificial intelligence: from monolithic large language models toward distributed multi-agent systems in which specialized LLM-powered agents collaborate, negotiate, and supervise one another [1]. Financial decision clusters, collaborative research platforms, industrial digital twins, urban traffic orchestration, and autonomous fleet coordination increasingly adopt multi-agent topologies because they decompose complex tasks, isolate failure domains, and exploit heterogeneous capabilities that no single model instance possesses [1,2]. Industry surveys document hundreds of deployed frameworks and a rapidly consolidating engineering practice around role specialization, tool use, and inter-agent communication protocols [1].

This migration, however, has exposed a layer of systemic pathologies that single-model engineering never confronted. An empirical failure taxonomy constructed from more than 1,600 annotated multi-agent traces identifies fourteen recurring failure modes spanning specification errors, inter-agent misalignment, task verification failures, and malformed coordination [3]. Four of these pathologies dominate production incident reports. First, decision oscillation: agents locked in zero-sum bargaining revise proposals indefinitely without converging, consuming compute while degrading output quality. Second, hallucination propagation: a fabricated fact accepted by one agent becomes premise material for its peers, converting an isolated generation error into a system-wide belief [4,5]. Third, long-horizon drift: systems scheduled for uninterrupted 24/7 operation accumulate small biases that individual checkpoints fail to remove, eventually departing from their operating envelope. Fourth, value boundary erosion: agents with growing autonomy progressively overstep policy constraints that static rule engines were expected to enforce.

The economic weight of these pathologies is worth making explicit, because it explains why they resist the incremental fixes that individually look sufficient. Oscillation is paid in latency and compute: an unconverged negotiation loop burns inference budget on every iteration while the user waits. Propagation is paid in trust: a single fabricated statistic that survives into a client deliverable can cost more reputation than the entire system earns in a quarter, and the propagation dynamic makes the cost super-linear in system size, since each additional agent multiplies the surfaces on which an error can land. Drift is paid in operations: deployments that would otherwise run unattended accumulate silent deviations that eventually demand human firefighting at inconvenient hours. Boundary erosion is paid in liability: an agent action that oversteps a compliance envelope converts a technical defect into a legal event. A runtime

architecture that addresses these four costs jointly, rather than one at a time, is therefore not a luxury but the precondition for the multi-agent economics to close.

Mainstream mitigations address these pathologies only partially. Instruction-dispatch orchestrators and greedy task routers resolve shallow coordination but carry no representation of inter-agent conflict. Reinforcement learning from human feedback (RLHF) and its variants align a model to human preferences during training but operate as an external patch that cannot inspect the runtime interaction between autonomous agents [6,7]. Self-play and adversarial training inject conflict productively during training, yet their zero-sum objective incentivizes a winner rather than a stable collective, and their convergence behavior remains fragile in practice [8,9]. Distributed consensus protocols guarantee agreement on shared state but presuppose correct components; they silently replicate a hallucinated value across replicas. What is missing is a runtime architecture in which conflict, correction, and control are designed as a single convergent loop rather than bolted together post hoc.

This paper addresses that gap. We propose the Recursive Adversarial Engine (RAE), a stability-oriented runtime architecture for LLM-based multi-agent systems. RAE rests on three design propositions: (i) bounded, non-destructive adversarial interaction is the principal source of error-detection signal in a closed multi-agent system; (ii) single-pass verification cannot eliminate systematic bias, and recursive, layered correction is necessary for convergence; (iii) long-run stability arises from dynamic checks and balances among agents rather than from centralized static control. These propositions are realized in a four-layer reference architecture—adversarial verification, recursive correction, checks-and-balances scheduling, and steady-state output—each specified at the level of modules, data flow, and interface contracts so that it can be directly implemented on contemporary agent frameworks.

The contributions of this paper are fourfold. First, we formulate a coherent conceptual framework that converts the three propositions into an implementable architecture, with explicit definitions of the deviation signal, the recursive correction operator, and the scheduling policy (Section 3). Second, we characterize the convergence behavior of the correction loop through a Lyapunov-style stability argument, identifying the conditions under which recursive correction decreases system deviation and the mechanisms that guard against its failure modes (Section 3.7). Third, we delineate the paradigm difference between RAE and four incumbent families—self-play, multi-agent debate, alignment pipelines, and consensus protocols—through a structured qualitative comparison (Section 4.5). Fourth, we analyze the engineering implications of the architecture for high-stakes deployment, long-horizon robustness,

auditability, and middleware integration, and we state its limitations candidly: RAE is a conceptual framework with a formal skeleton, and its claims require empirical validation on benchmark workloads before adoption in safety-critical settings (Section 4.6).

The remainder of the paper is organized as follows. Section 2 reviews related work in five strands. Section 3 develops the RAE architecture and its stability analysis. Section 4 discusses engineering significance, comparisons, and limitations. Section 5 concludes.

2. Related Work

2.1. Adversarial Training and Self-Play

Adversarial formulation has repeatedly proven to be a powerful engine of capability. Generative adversarial networks (GANs) established that two networks locked in a minimax contest can implicitly learn rich distributions, and successive refinements—Wasserstein objectives, gradient penalty, and two-time-scale updates—steadily improved the stability of the underlying game [8,10,11,12]. Nevertheless, the GAN literature is equally a catalogue of pathologies: non-convergence, mode collapse, and oscillation between generator and discriminator persist even in well-tuned systems, and formal guarantees are limited to local Nash equilibria under restrictive assumptions [9,12]. Self-play curricula in game playing share the same structure: an agent improves by contesting copies of itself until the opponent pool saturates. Both lines teach an essential lesson that RAE takes seriously—adversarial pressure is a potent source of learning signal, but unbounded or mis-objectived conflict destabilizes the very system it is meant to improve. The zero-sum objective that serves a two-player training game is counterproductive when the objective is a cooperative, long-lived multi-agent deployment: it incentivizes a victor rather than a stable equilibrium of the collective.

A second, less discussed distinction concerns the lifetime of the adversarial relationship. In training games, conflict is a phase: once the opponent pool saturates, adversarial pressure is withdrawn and the artifact ships. In a deployed multi-agent system there is no post-training moment—the collective runs for months, its membership changes, its task distribution drifts, and the adversarial relationship, if it is to keep detecting errors, must be a permanent, institutionalized feature of the runtime rather than a curriculum that ends. None of the incumbent training-time paradigms were designed for permanent conflict under deployment constraints: budgets, auditability, and graceful degradation are not concerns a GAN ever had. The architecture proposed in this paper takes permanent, bounded conflict as its starting design condition.

2.2. Multi-Agent Debate and LLM Collectives

Within the LLM era, the closest neighbors of RAE are debate-style collectives. Irving et al. proposed AI safety via debate, in which two agents argue opposing positions and a judge model evaluates the exchange, with the aim of surfacing flaws that obfuscated single-model outputs would hide [13]. Du et al. demonstrated that having multiple LLM instances propose and critique successive answers improves factuality and reasoning on arithmetic and strategic tasks [2]. Framework surveys catalog a growing zoo of role-played collectives—programmer, critic, planner, executor—that operationalize division of labor with conversational protocols [1]. Two structural limitations recur across these systems. First, the debate terminates by design after a fixed number of rounds, with a judge or majority vote as the arbiter; there is no mechanism that distinguishes residual disagreement caused by genuine task difficulty from disagreement caused by persistent hallucination, and no guarantee that the accepted answer is internally consistent across the collective. Second, participants in the debate are peers with equal standing and static permissions; a participant that repeatedly fabricates evidence exerts the same influence in round ten as in round one. RAE inherits the adversarial-interview insight but embeds it in a recursive loop with per-agent accountability and graded intervention.

2.3. Alignment Pipelines and Safety Layers

A parallel literature hardens individual models against unsafe or unfaithful behavior. RLHF established preference learning from human comparisons as the dominant alignment signal [6]; Constitutional AI replaced part of the human loop with principle-guided self-critique [7]; automated red-teaming scales adversarial probing of model behavior before deployment [14,15]. Safety agendas across the field enumerate concrete hazards—negative side effects, reward hacking, scalable oversight—and unsolved problems that reservation must confront [16-18]. These pipelines are indispensable, but they are properties of a model at training time, not of a fleet at run time. Hallucination surveys document that no current mitigation eliminates fabrication, and that residual errors are precisely the substrate on which multi-agent propagation operates [4,5]. RAE is complementary to this line: it assumes aligned component models and targets the emergent failure modes of their interaction—oscillation, propagation, drift—which training-time alignment cannot reach.

The complementarity has a practical corollary for system builders. Because alignment quality and runtime stability are different properties, neither can substitute for the other in a deployment argument: a perfectly aligned model can still participate in an oscillating collective, and an impeccably scheduled collective still inherits the fabrication tendencies of its members. The

engineering consequence is that acceptance criteria for multi-agent systems need two columns—one for component-level alignment evidence and one for system-level stability evidence—and much of the current difficulty in certifying multi-agent deployments reflects the absence of a standard form for the second column. Providing a candidate for that form is among the intentions of the architecture developed below.

2.4. Stability Theory and Self-Regulating Systems

The third strand supplies RAE's analytical vocabulary. Lyapunov stability theory provides the canonical condition under which a dynamical system returns to an equilibrium: the existence of a scalar energy-like function that decreases along all trajectories [19-21]. Autopoietic and self-organizing systems research describes how living and computational systems maintain identity through continuous internal reorganization, and how apparently chaotic collectives can exhibit phase transitions between order and disorder [22,23]. Recursion itself has been studied since the foundational work of computability theory, with fixed-point theorems guaranteeing that self-referential definitions admit unique stable interpretations under contraction conditions [24,25]. Hofstadter's treatment of strange loops further illuminates how tangled hierarchies resolve into stable levels [26]. RAE imports two ideas from this tradition: the requirement that its correction loop admit a decreasing deviation functional, and the insistence that stability be an emergent property of mechanism design rather than an externally imposed constraint.

2.5. Positioning

Across the four strands, no existing framework simultaneously provides bounded adversarial signal generation, recursive correction with convergence accountability, and dynamic checks-and-balances control as one integrated runtime. Self-play and GANs supply conflict without convergence guarantees suitable for deployment; debate supplies signal without enforcement; alignment supplies constraint without runtime visibility; consensus supplies consistency without error detection. RAE is positioned as the integration layer that closes this gap for LLM-based multi-agent systems.

3. The Recursive Adversarial Engine

3.1. Design Rationale: Three Foundational Propositions

RAE's architecture is derived from three propositions about the nature of error and stability in autonomous multi-agent systems.

Proposition 1: bounded adversarial interaction is the principal negative-entropy source of a

closed multi-agent system. A collective of mutually agreeable agents constitutes a structurally closed loop in which errors circulate unchallenged; politeness gradients suppress dissent exactly where dissent is most informative. Controlled, boundary-respecting adversarial behavior—cross-examination of proposals, mutual consistency interrogation, compliance scanning—exposes latent defects, identifies decision bias, and escapes local optima, supplying the system with corrective signal it cannot generate internally [23]. The qualifier "bounded" is load-bearing: unbounded conflict degrades into the oscillation and attrition documented in the adversarial-training literature [9,12], so adversarial exchanges must be licensed, scoped, and terminated by protocol rather than by agent discretion.

Proposition 2: single-pass verification cannot eradicate systematic deviation; recursive iteration is necessary for convergence. Hallucination and value drift are characterized by concealment and propagation: an error removed from one layer re-enters through another, and a correction applied without re-inspection introduces fresh bias [4,5]. Single-shot cross-checking therefore only trims surface errors. Multi-round recursion—correct locally, verify globally, re-tune locally, confirm global consistency—drives deviation down layer by layer, provided the loop is contracting rather than divergent. This is the same insight that underlies fixed-point semantics of self-referential definitions [25] and that motivates retaining the full recursion path as first-class system state.

Proposition 3: steady-state balance is achieved by dynamic checks and balances, not by static control. Fully centralized control suffocates agent autonomy and re-creates the single point of failure that multi-agent design was meant to remove; unconstrained autonomy dissolves the system into the oscillation of Proposition 1's negative case. The viable middle is mechanism design: permissions, decision weights, and intervention thresholds that adjust continuously to observed agent behavior, so that the system trades autonomy against safety along a controlled frontier [22]. RAE operationalizes this as a scheduling layer with graded authority and graduated circuit-breaking.

The three propositions are not independent axioms but a division of labor. Proposition 1 concerns signal: it explains where corrective information comes from in a closed system, namely from licensed conflict among the agents themselves. Proposition 2 concerns dynamics: it explains how that information must be consumed—recursively, with re-verification—if it is to eliminate rather than merely relocate error. Proposition 3 concerns control: it explains how the first two can operate for the long run without either freezing the system or dissolving it, by making authority itself a state variable that responds to evidence. The four-layer architecture of

Sections 3.3 through 3.6 realizes this division directly: Layer 1 instantiates the signal, Layer 2 the dynamics, Layer 3 the control, and Layer 4 the closure that turns each run's experience into the system's own prior.

3.2. Formal Problem Statement

Consider a multi-agent system whose collective state at discrete time t is a vector s_t comprising agent outputs, intermediate reasoning artifacts, and interaction logs. Let $D: S \rightarrow R_{\geq 0}$ be a deviation functional that aggregates, over the state, the four pathology dimensions introduced in Section 1: factual inconsistency with grounded evidence, cross-agent disagreement, policy-boundary violation, and unresolved conflict. The runtime goal of RAE is to transform an arbitrary operating state into a steady state—a state $s*$ satisfying:

$$|D(s^*)| \leq \varepsilon \tag{1}$$

where ε is the application's tolerance vector, while preserving agent autonomy and task progress. An idealized one-step rendering of the engine's action is the composed operator $s_{t+1} = C(B(s_t))$, where B denotes bounded adversarial verification (deviation measurement) and C denotes recursive correction with checks-and-balances scheduling. The architecture question is how to realize B and C so that the composed map is contracting in the region of interest, i.e., so that there exists a comparison functional $V(s) = \|D(s)\|^2$ whose value decreases along successive iterations:

$$V(s_{t+1}) \leq \gamma \cdot V(s_t), \quad 0 < \gamma < 1 \tag{2}$$

until the stopping condition (1) is met or a safety override fires. Equation (2) is the interface between RAE's engineering layers and classical stability theory [19,21]; Section 3.7 discusses which architectural mechanisms make the inequality plausible and what happens when it fails.

3.3. Layer 1: Adversarial Verification

The verification layer is the engine's sensory apparatus. It deploys four cooperating modules. The cross-adversarial review module pairs agents so that each decision proposal, output artifact, and reasoning path is examined by an independent peer tasked with falsification rather than agreement; the examiner's objective is to construct the strongest refutation, and its reward excludes agreement outcomes, inverting the sycophancy gradient of cooperative collectives. The proposal cross-check module enforces mutual inspection of decision proposals for internal contradiction, unsupported premises, and inferential gaps. The fact-consistency module anchors claims against available grounded sources and against the system's own verified memory,

flagging fabricated or unverifiable content [4]. The boundary-compliance scanner screens every candidate action against the applicable policy envelope—legal constraints, ethical guidelines, permission scopes—classifying borderline behavior for escalation rather than silent passage [7].

The layer's output is a structured deviation inventory: each finding carries a deviation type, affected agents, evidence pointers, a risk grade, and a propagation estimate. This structure is deliberate. First, it converts diffuse conflict into a machine-actionable signal, realizing the transformation from unregulated dispute to controlled verification data. Second, it makes the deviation functional D of Section 3.2 computable at runtime rather than merely definable. Third, it is the primitive on which auditability downstream depends: every accepted deviation becomes a traceable event in the system's evidentiary record.

3.4. Layer 2: Recursive Correction

The correction layer consumes the deviation inventory and operates as a bounded recursion. Each cycle executes four stages: local repair, in which the owning agent revises the flagged artifact under the examiner's findings; global association check, in which a coordination pass re-verifies every state element downstream of the repair, since the deviation inventory's propagation estimate defines the re-verification frontier; secondary local tuning, in which residual inconsistencies surfaced by the global check are patched; and global consistency confirmation, in which the cycle terminates only when the deviation functional over the affected region falls below the local tolerance. Crucially, each recursion round re-inspects the previous round's repairs, so corrections cannot silently introduce new deviations of the kind they removed.

Three design decisions distinguish this loop from a naive retry. First, termination is measured, not counted: the loop exits on deviation collapse (equation (2) satisfied locally), on progress stagnation, or on budget exhaustion—not after a fixed number of chat turns, which is the debated norm in debate collectives [2]. Second, the complete recursion path—the inventory that triggered the cycle, each intermediate repair, each re-verification verdict—is retained as immutable system state. This record is the raw material of interpretability (Section 4.3) and the substrate for post hoc analysis of correction quality. Third, the loop is designed with contraction in mind: by reusing the same verifier family across rounds and requiring monotone inventory shrinkage, the architecture aims at the fixed-point condition under which self-referential correction settles to a unique consistent state [24,25]. Contraction is an engineering target to be monitored, not a guarantee bestowed by design; Section 3.7 makes the failure modes explicit.

3.5. Layer 3: Checks-and-Balances Scheduling

The scheduling layer converts verification outcomes into control actions, implementing Proposition 3's dynamic checks and balances. It maintains, per agent, a continuously adjusted decision weight and permission scope. Weight adaptation is evidence-driven: an agent whose outputs repeatedly attract high-risk deviations, or whose adversarial exchanges exceed the behavioral envelope, has its temporary weight reduced, which degrades its influence on collective decisions and routes its proposals through strengthened verification. Permission adjustment is symmetrical—an agent with a sustained clean record earns wider autonomy—which preserves the incentive for reliable behavior rather than merely punishing deviation.

Conflict arbitration ranks concurrent disagreements by a fixed lexicographic order: safety impact first, responsibility tier second, decision weight third. This ordering ensures that when, for example, a planning agent and a compliance agent deadlock, the compliance constraint prevails regardless of the planners' seniority. The layer also owns the graded circuit breaker. A circuit breaker is triggered when global conflict rate, deviation density, or recursion stagnation crosses its threshold; the response is graduated—throttling non-critical agents, freezing discretionary tasks, rolling back to the last verified state—rather than a binary halt, so that core mission progress and safety invariants are protected at minimal operational cost. Circuit breaking deliberately mirrors established practice in control systems, where protective action must be fast, layered, and reversible [20,21].

fast, layered, and reversible [20,21].

The graded design also reconciles two requirements that a binary kill-switch cannot jointly satisfy: containment granularity and recoverability. Because interventions attach to agents and task classes rather than to the system as a whole, a defecting summarizer is throttled while a serving path continues; and because the response to any trigger is a sequence—throttle, freeze, roll back to the last certified state—an intervention that proves unnecessary is undone by progressing back down the same sequence. Trigger thresholds themselves are adaptive at slow timescales: sustained near-threshold behavior in a task class tightens that class's thresholds preemptively, which is the mechanism through which the scheduling layer contributes to the long-horizon drift control discussed in Section 4.2.

3.6. Layer 4: Steady-State Output

The output layer integrates the corrected state with the scheduling layer's verdicts and emits the system's deliverables. Its obligations are threefold. It produces the final decision or artifact, certified as globally consistent, factually grounded, and policy-compliant with respect to the

deviation inventory. It produces the run's evidentiary bundle: the recursion verification report, the conflict-resolution report, and a robustness score that summarizes deviation trajectory, correction depth, and circuit-breaker activity during the run. And it closes the loop at the temporal scale of a single run: the evidentiary bundle feeds weight adaptation at the scheduling layer and rule refinement at the verification layer, so that each run's correction experience becomes the next run's prior. Table 1 summarizes the four layers, their core modules, and the property each guarantees.

Table 1. The four-layer architecture of the Recursive Adversarial Engine.

Layer	Core modules	Primary input	Primary output	Guarantee contributed
Adversarial verification	Cross-adversarial review; proposal cross-check; fact-consistency; boundary-compliance scan	Agent proposals, outputs, reasoning paths	Structured deviation inventory with risk grades	Detectability of hallucination, conflict, and violation
Recursive correction	Local repair; global association check; secondary tuning; consistency confirmation	Deviation inventory	Repaired state; immutable recursion path	Monotone deviation reduction toward steady state
Checks-and-balances scheduling	Weight adaptation; permission scoping; conflict arbitration; graded circuit breaking	Deviation statistics; conflict telemetry	Adjusted weights, scopes, interventions	Dynamic equilibrium of autonomy and safety
Steady-state output	Result integration; report generation; robustness scoring	Corrected state; scheduling verdicts	Certified deliverables; audit bundle; feedback to Layers 1 and 3	Global consistency, traceability, run-level closure

3.7. Convergence Behavior and Failure Modes

The stability argument for RAE is comparative and conditional, and it is worth stating precisely what is and is not claimed. What is claimed: the architecture instantiates the structure of a stabilizing feedback system, in the sense of Lyapunov's classical criterion [19]. The deviation functional $V(s) = \|D(s)\|^2$ of equation (2) plays the role of an energy function; the verification layer provides continuous measurement of V; the correction layer supplies a repair operator whose cycle semantics require inventory shrinkage before termination; and the scheduling layer bounds the influence of any component whose behavior would drive V upward. Under two assumptions—(A1) verifier judgments are more often right than the agents they

judge, so that each repair reduces the true deviation in expectation; and (A2) correction resources are sufficient to reach the affected region before propagation outruns repair—each recursion cycle is expected to contract V, and the system drifts toward the steady state of equation (1) [20,21]. Assumption A1 is the epistemic load-bearing wall: it is supported operationally by grounding the fact-consistency module against external evidence rather than agent consensus [4], and by the lexicographic precedence of compliance arbitration, but it is an empirical property of a deployment, not a theorem.

The failure modes are correspondingly explicit. If verifiers systematically err (A1 violated), the loop can converge to a confidently wrong steady state; the audit bundle is designed precisely to make this detectable post hoc, and weight adaptation shrinks the influence of agents whose findings are later overturned. If propagation outruns correction (A2 violated), the graded circuit breaker freezes discretionary activity before the deviation spreads system-wide, trading availability for integrity. If adversarial pressure exceeds the behavioral envelope, oscillation re-emerges; the boundary scanner and conflict-rate threshold catch it and throttle the exchange. Finally, the engine itself consumes compute; recursion depth and arbitration frequency are governed by explicit budgets so that stability expenditures remain bounded. These mechanisms do not prove convergence—they bound the modes of divergence, which for a deployable system is the practically meaningful guarantee [9,12].

practically meaningful guarantee [9,12].

It is useful to relate this stability picture to the detect-respond discipline of mature reliability engineering. The verification layer implements detection with coverage obligations—every certified output must carry a deviation inventory, so undetected deviation can pass only through verifier failure, never through absence of testing. The correction and scheduling layers implement response with measured escalation, and the audit bundle implements post-incident analysis with complete evidence. The mapping is deliberate: it means RAE's operational health can be monitored with the same service-level vocabulary—detection latency, correction success rate, breaker activation rate—that production systems already use, and its stability claims are therefore falsifiable through ordinary operational telemetry rather than through bespoke evaluation alone [16,17].

The fourth mechanism addresses the cost side of the ledger. Recursive correction multiplies inference expenditure precisely when the system is least healthy, and an unpriced loop can turn an incident into an incident plus a budget overrun. RAE therefore attaches budgets to recursion at two granularities: per-cycle token and call budgets enforced by the scheduling layer, and per-

run cumulative budgets whose exhaustion converts the run from corrective to conservative mode—delivering with an explicit uncertified annotation rather than consuming unlimited resources. The annotation pathway preserves the honesty property of Section 3.6 under scarcity: when the engine cannot afford to verify, it says so on the artifact instead of pretending the risk away.

4. Engineering Implications and Discussion

4.1. Safety Envelopes for High-Stakes Domains

The first deployment dividend of the architecture appears in high-reliability domains: quantitative finance, clinical decision support, legal drafting, and public-sector approval workflows. In these settings the binding constraint is not capability but error tolerance: regulatory regimes demand that decision error, fabrication, and conflict rates remain inside auditable bounds [16,18]. RAE's contribution to this constraint is architectural rather than statistical. The deviation inventory gives risk a runtime representation with explicit grades; the recursive correction loop drives residual deviation toward the tolerance envelope before output certification; and the steady-state output refuses certification when the envelope is not met, converting a silent quality failure into an explicit, logged refusal. The comparison with incumbent pipelines is direct: RLHF-style alignment shapes component models before deployment [6,7], but the acceptance decision at run time still requires evidence that this particular output is deviation-free; RAE supplies exactly that evidence chain.

Concretely, consider investment-research report generation under a suitability regime. The verification layer's boundary scanner enforces that every performance claim carries a computation basis and that restricted-product language cannot reach the draft stage; the recursion loop requires that a corrected valuation be re-derived, not merely re-worded; and the steady-state output certifies the report together with its deviation inventory, so the compliance function receives, for each document, the list of what was found and what was corrected. From the regulator's perspective the difference is qualitative: instead of a vendor attestation that models were aligned in training, the institution can produce per-artifact process evidence—finding, repair, re-verification—of the kind audit standards already presuppose [6,18].

4.2. Long-Horizon Robustness

The second dividend concerns systems scheduled for uninterrupted operation: industrial IoT coordination, urban scheduling, communication network optimization. Long-horizon deployments fail not through any single catastrophic error but through drift—small biases that

pass each individual checkpoint yet compound across operating hours, a dynamic reminiscent of self-organized criticality in which accumulating stress releases through avalanches of varying size [23]. RAE counteracts drift through two mechanisms with complementary time constants. At the fast timescale, every run's recursion loop removes deviation down to tolerance before certification, so each delivered artifact starts from a verified base. At the slow timescale, the evidentiary bundles accumulate into the scheduling layer's statistics, and sustained patterns—rising correction depth for a given agent, increasing conflict density in a given task class—trigger preventive weight and permission adjustments before thresholds are breached. The engineering claim is that this converts drift from an unbounded random walk into a regulated process with observable early-warning indicators, thereby reducing emergency intervention and manual oversight cost. Again, the claim is architectural; validating it requires longitudinal field measurement, which we flag in Section 4.6.

which we flag in Section 4.6.

A metropolitan scheduling cluster makes the two-timescale mechanism concrete. Hourly, signal-timing and transit-consolidation proposals pass through adversarial verification against live sensor feeds; a misparsed congestion reading is caught as factual inconsistency before it contaminates a corridor plan, corrected in one recursion cycle, and recorded. Weekly, the scheduling layer aggregates the correction record: if one perception agent's findings are increasingly overturned, its weight decays before its errors compound; if a task class shows rising correction depth, its verification rules are tightened. Neither timescale requires the system to halt. The design intent is that drift is consumed continuously at the run level and structurally at the season level, so the accumulation dynamics that produce avalanche-style failures [23] never find an unmonitored interval in which to operate.

4.3. Interpretability and Auditability

The third dividend addresses the black-box problem that currently blocks adoption in audited industries. Interpretability is usually pursued by explaining a model's internal computation, an approach whose outputs are themselves unverifiable generations. RAE relocates interpretability from model introspection to process evidence: because every deviation finding, every recursive repair, every arbitration decision, and every circuit-breaker activation is written to the immutable run record, the system's behavior is reconstructible as a causal narrative without interpreting any neural internals. An auditor can answer the questions that regulation actually asks—why was this output certified, what deviations were found, who intervened, what was overridden—entirely from the record [7,14]. The robustness score attached to each run gives

oversight bodies a single comparable metric across deployments. This evidence-first stance parallels the direction of the AI-safety field toward scalable oversight and evaluation: rather than demanding that a model explain itself, demand that the process leave evidence [17,18].

demand that the process leave evidence [17,18].

The practical shape of such an audit is worth sketching. Asked why a loan-advisory output was certified, the institution does not interrogate a model; it replays the run record: the deviation inventory attached to the certified state, the two high-risk findings (an unsupported income inference and a policy-boundary approach), the recursion path that repaired the first and re-derived the second, the arbitration decision that resolved a disagreement between the compliance and advisory agents, and the robustness score contextualized against the deployment's history. Every element is a logged system event rather than a generated explanation, so the narrative is reconstructible from evidence that existed before the question was asked—an epistemically sturdier position than post hoc rationalization [14].

4.4. Middleware Integration and Ecosystem Effects

The fourth dividend is integrative. RAE is specified as a runtime layer with narrow interfaces—deviation inventories in, certified states and verdicts out—that do not presuppose a particular agent framework, model provider, or communication protocol. This positions the engine as middleware: heterogeneous LLM agents, open-source agent frameworks, and domain-specific modules can be attached beneath the verification layer without modification, and receive scheduling and certification services from above. The practical consequence is a standardization path for organizations that currently face high bespoke integration costs when composing multi-agent systems from diverse components [1,3]. Ecosystem-wise, uniform deviation semantics is the deeper prize: once verification findings, arbitration records, and robustness scores share a schema, cross-organizational benchmarking of reliability becomes possible in a way that isolated deployments have not supported, and failure knowledge can accumulate across the community rather than within single engineering teams [3].

The middleware posture also carries a governance interface. Because deviation grades, arbitration orders, and circuit-breaker thresholds are configuration rather than code, an adopting organization encodes its own risk appetite—what counts as high risk, which constraints rank above throughput, when availability yields to integrity—without modifying the engine. This separation matters operationally: risk appetite differs across deployments of the same architecture, and hard-wiring any single appetite into engine code would either over-constrain permissive domains or under-protect regulated ones [16,18].

Standardization of the interface is likewise what makes the ecosystem claim concrete. If deviation inventories, arbitration records, and robustness scores converge on shared schemas, then third parties can build cross-framework dashboards, benchmark suites can compare reliability across agent frameworks on equal terms, and incident reports become machine-ingestible by the community at large. The history of infrastructure software suggests that such shared observational schemas—not any single framework's superiority—are what eventually consolidate a fragmented ecosystem, because they convert every participant's operational experience into a public good that all participants, including the framework vendors, consume [1].

4.5. Comparison with Mainstream Paradigms

Table 2 positions RAE against the four incumbent families along five dimensions: the objective of adversarial interaction, the correction strategy, the control structure, the convergence story, and the characteristic failure each leaves unaddressed. The comparison substantiates the paradigm claim made in Section 1: existing frameworks each solve a genuine sub-problem—training signal, factuality improvement, behavioral constraint, or state agreement—and RAE's contribution is the integration of all four concerns under a single runtime contract oriented to collective stability.

Three contrasts deserve emphasis. Against self-play, RAE rejects the zero-sum premise: adversarial energy exists to serve collective convergence, not to crown a victor, which is why examiner incentives exclude agreement outcomes while the collective objective remains cooperative [8,9]. Against debate collectives, RAE replaces round-counting with measured termination and static peer standing with evidence-driven weight adaptation, so that a habitually fabricating participant cannot steer the collective indefinitely [2,13].

A further engineering pain point in deployed collectives deserves mention: protocol fragility under participant turnover. Debate-style systems typically assume a fixed roster and a well-formed transcript; in production, agent versions are redeployed mid-conversation, contexts are truncated by budget, and participants join or leave. Framework surveys record malformed interactions and unreliable tool behavior among the dominant failure categories observed in practice [1,3]. RAE's verification layer is comparatively robust to churn because its unit of trust is the graded finding rather than the participant: a newly deployed agent inherits verification treatment from its role and history class, not from conversational seniority, and a redeployed agent's unverified proposals simply re-enter the inventory like any others. Against alignment and consensus, RAE operates inside the runtime window where emergent interaction failures

live, detecting what training-time pipelines cannot see and reconciling what consensus cannot question [6,7].

Table 2. Qualitative comparison of RAE with mainstream multi-agent and alignment paradigms.

Paradigm	Representative methods	Adversarial objective	Correction strategy	Control structure	Residual gap
Self-play / adversarial training	GANs and variants [8,10,11,12]; self-play curricula	Zero-sum; a winner must emerge	Training-time optimization	Implicit in game dynamics	Deployment-time stability unaddressed; oscillation persists [9]
Multi-agent debate	Debate collectives [2,13]	Improve answer quality via critique	Fixed-round judging / voting	Static peer roles	No per-agent accountability; residual disagreement unclassified
Alignment pipelines	RLHF [6]; Constitutional AI [7]; red-teaming [14,15]	Shape model behavior to preferences	Pre-deployment training	External to runtime	Runtime interaction failures out of scope
Distributed consensus	Paxos-family protocols	None (assumes correct components)	State reconciliation	Quorum rules	Replicates corrupted content; no error detection
RAE (this work)	Four-layer architecture (Section 3)	Bounded, non-zero-sum verification signal	Recursive correction with convergence monitoring	Dynamic checks and balances	Empirical validation pending (Section 4.6)

4.6. Limitations and Open Challenges

Candor about scope is a design obligation for a framework paper. RAE as presented here is a conceptual architecture with a formal skeleton; no claims in this paper rest on experimental measurement. Six limitations bound its current status. First, empirical validation is pending: the architecture's central predictions—that measured recursive correction reduces deviation monotonically in realistic workloads, and that weight adaptation degrades influence of unreliable agents fast enough to matter—require controlled studies on multi-agent benchmarks and the failure taxonomies now available [3]. Second, the deviation functional D aggregates heterogeneous dimensions (factual, consistency, compliance, conflict) whose weights are inherently contestable; choosing them is a governance decision as much as a technical one. Third, computational overhead is real: adversarial verification and multi-round recursion multiply inference cost, and the cost–stability frontier must be mapped per application class before deployment commitments. Fourth, verifier reliability is the epistemic foundation (assumption A1, Section 3.7), and correlated verifier–agent failures—a shared base model

replicating a blind spot—could defeat the architecture's core redundancy; verifier diversity is therefore not an optimization but a safety requirement. Fifth, adversarial dynamics introduce their own risks: examiner collusion, reward gaming of deviation grades, and strategic behavior around circuit-breaker thresholds are all plausible and require mechanism-design countermeasures of the kind the red-teaming literature studies [14,15]. Sixth, the Lyapunov-style analysis is an engineering analogy with explicit assumptions, not a theorem; a formal treatment—deviation functionals with approximation guarantees, contraction conditions for LLM-mediated repair operators—remains open, as does the interface to training-time alignment so that runtime and pre-deployment mechanisms reinforce rather than duplicate each other [17,18]. These challenges define the experimental agenda that must precede any safety-critical adoption.

must precede any safety-critical adoption.

A validation agenda can nonetheless be stated precisely, and its feasibility is itself an argument for the architecture. On benchmarks with constructable ground truth, the deviation functional's measurement validity can be established against annotated failures of the kind the MAST taxonomy provides [3]; recursion efficacy is testable as inventory shrinkage per cycle under ablations that disable individual layers; weight-adaptation dynamics are observable as influence curves for seeded unreliable agents; and overhead is measurable as the cost–stability frontier across recursion budgets. Each experiment maps onto an architectural mechanism rather than onto an emergent property, which is what distinguishes evaluating an architecture from evaluating a prompt arrangement, and each produces telemetry that downstream deployments would monitor in production anyway [17].

5. Conclusion

This paper proposed the Recursive Adversarial Engine, a stability-oriented runtime architecture for large language model multi-agent systems, motivated by four production-scale pathologies that incumbent orchestration, debate, alignment, and consensus frameworks each leave partially unaddressed: decision oscillation, hallucination propagation, long-horizon drift, and value-boundary erosion. RAE's design rests on three propositions—bounded adversarial interaction as the system's error signal, recursive correction as its error-elimination mechanism, and dynamic checks and balances as its stability guarantee—realized in a four-layer architecture specified at module and interface level: adversarial verification, recursive correction, checks-and-balances scheduling, and steady-state output.

The paper's analysis yields three takeaways. First, conflict and stability are not opposites in multi-agent design: the architecture converts adversarial pressure into a controlled verification signal and therefore into convergent correction, inverting the zero-sum premise of self-play while retaining its generative energy [8,9]. Second, stability must be engineered as an emergent property of mechanism design—measured termination, evidence-driven weights, graded circuit breaking—rather than asserted by static rules, which connects runtime multi-agent engineering to the classical vocabulary of stability theory [19-21]. Third, auditability is an architectural by-product that RAE obtains nearly for free: because every deviation, repair, arbitration, and intervention is recorded, process evidence replaces model introspection as the substrate of oversight [17,18].

The framework's limitations are stated without hedging in Section 4.6, and they define the road ahead: empirical evaluation against multi-agent failure taxonomies, formalization of deviation functionals and contraction conditions, cost–stability characterization across application classes, and verifier-diversity requirements. The destination justifies the effort. As multi-agent systems assume responsibility for decisions in finance, medicine, infrastructure, and government, the engineering question that matters is no longer whether agents can collaborate, but whether their collaboration can be made to converge, to remain within its value boundaries, and to account for itself. This paper contributes an architecture aimed at exactly that question.

References

[1] Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N. V., Wiest, O., & Zhang, X. (2024). Large language model based multi-agents: A survey of progress and challenges. arXiv preprint arXiv:2402.01680.

[2] Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., & Mordatch, I. (2024). Improving factuality and reasoning in language models through multiagent debate. In Proceedings of the 41st International Conference on Machine Learning (ICML).

[3] Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., & Song, D. (2025). Why do multi-agent LLM systems fail? arXiv preprint arXiv:2503.13657.

[4] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), 1-35.

[5] Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., & Liu, T. (2023). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. arXiv preprint arXiv:2311.05232.

[6] Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. In Advances in Neural Information Processing Systems (Vol. 30).

[7] Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., et al. (2022).

Constitutional AI: Harmlessness from AI feedback. arXiv preprint arXiv:2212.08073.
[8] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Advances in Neural Information Processing Systems (Vol. 27).
[9] Mescheder, L., Geiger, A., & Nowozin, S. (2018). Which training methods for GANs do actually converge? In Proceedings of the 35th International Conference on Machine Learning (ICML).
[10] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In Proceedings of the 34th International Conference on Machine Learning (ICML).
[11] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of Wasserstein GANs. In Advances in Neural Information Processing Systems (Vol. 30).
[12] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In Advances in Neural Information Processing Systems (Vol. 30).
[13] Irving, G., Christiano, P., & Amodei, D. (2018). AI safety via debate. arXiv preprint arXiv:1805.00899.
[14] Ganguli, D., Hernandez, D., Lovitt, L., Askell, A., Bai, Y., Chen, A., et al. (2022). Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. arXiv preprint arXiv:2209.07858.
[15] Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Gladstone, A., Irving, G., & Christiano, P. (2022). Red teaming language models with language models. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP).
[16] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565.
[17] Hendrycks, D., Carlini, N., Schulman, J., & Steinhardt, J. (2021). Unsolved problems in machine learning safety. arXiv preprint arXiv:2109.13916.
[18] Russell, S. (2019). Human compatible: Artificial intelligence and the problem of control. Viking.
[19] Lyapunov, A. M. (1992). The general problem of the stability of motion. International Journal of Control, 55(3), 531-534.
[20] Slotine, J. J. E., & Li, W. (1991). Applied nonlinear control. Prentice Hall.
[21] Khalil, H. K. (2002). Nonlinear systems (3rd ed.). Prentice Hall.
[22] Varela, F. J. (1979). Principles of biological autonomy. North-Holland.
[23] Bak, P., Tang, C., & Wiesenfeld, K. (1987). Self-organized criticality: An explanation of the 1/f noise. Physical Review A, 38(1), 364-374.
[24] Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, s2-42(1), 230-265.
[25] Banach, S. (1922). Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. Fundamenta Mathematicae, 3, 133-181.
[26] Hofstadter, D. R. (1979). Gödel, Escher, Bach: An eternal golden braid. Basic Books.