AI-Intent: A Conceptual Modeling Framework for Accountable Multi-Agent AI Systems

Wolfgang Maass[1] and Iris Reinhartz-Berger[2]

[1] German Research Center for Artificial Intelligence (DFKI)
and Saarland University, Saarbrücken, Germany
wolfgang.maass@dfki.de
[2] University of Haifa, Haifa, Israel
iris@is.haifa.ac.il

Abstract. Existing agent-oriented conceptual modeling frameworks assume components with deterministic execution semantics. Applied to language model agents, whose outputs arise from probabilistic text generation governed by natural-language inputs, they lack three constructs that accountability requires: a binary *declaration* of each agent's legitimate action space; runtime *enforcement* of negative obligations by a component external to the agent; and treatment of inter-agent communication as *auditable evidence*.
To address these gaps, we introduce AI-Intent, a conceptual modeling framework organized into three pillars: ***Declaration*** encodes each agent's action boundary, decision rights, capabilities, and risk policies in *Mandates*; ***Enforcement*** evaluates every *Proposed Action* against the applicable *Mandate* through a *Compliance Agent* before delivery; ***Auditability*** derives a structured, durable *Accountability Trace* from every session. A reference implementation for private investment advisory under MiFID II is evaluated across 190 evaluation sessions: boundary violation containment averaged 97.9% with zero forced-pass occurrences, while audit trace completeness was found to depend on the instruction-following capability of the deployed model.

Keywords: Multi-Agent Systems · Conceptual Modeling · Language Model Agents · Accountability · Governance

1 Introduction

Language model-based agents are rapidly moving from research prototypes to production systems that support consequential decisions in finance, healthcare, and enterprise operations [23,27]. Unlike traditional software components, whose runtime behavior is fully determined by source code, they generate outputs through probabilistic text generation governed by natural language inputs, training data, and model parameters. When multiple such agents collaborate on joint recommendations, questions arise that existing conceptual models cannot answer: what each agent was supposed to do, whether it stayed within its scope, and why a particular output was produced.

Conceptual modeling frameworks for multi-agent systems have a rich history [28,8,24], modeling agents in terms of goals, capabilities, roles, and inter-agent dependencies. These frameworks share an assumption that their components have deterministic or rule-governed execution semantics – an assumption that does not hold for language model agents, giving rise to three gaps.

Gap 1: Intent boundary versus goal satisfaction. Traditional goal models, such as i^* and Tropos, treat goal satisfaction as graded [28]. For language model agents, the relevant construct is instead a binary *sortal boundary*: a request either falls within the agent's legitimate scope or it does not. An equity analyst agent recommending gold at 35% allocation has not partially satisfied its goal; it has transgressed its mandate scope.

Gap 2: Negative obligations as runtime-enforceable constraints. Normative frameworks and deontic logic provide well-developed accounts of obligation, permission, and prohibition [25,14,7], but assume agents that deliberate over norms before acting [9]. Language model agents may acknowledge a constraint and then violate it, or construct superficially plausible justifications for exceptions [17]. The modeling challenge is therefore to represent negative obligations in a form an *external* enforcement component can evaluate, rather than relying on agent self-compliance.

Gap 3: Communication as audit evidence. Existing agent communication models, such as FIPA-ACL [12], specify message syntax and semantics but treat communication logs as an optional engineering concern. Regulatory frameworks for AI systems, e.g., the EU AI Act [11] and MiFID II [10], require that every consequential decision be traceable to its inputs – making communication logging a *structural property* of the conceptual model, not an add-on.

To address these gaps, we propose AI-Intent, with one pillar per gap. ***Declaration*** introduces the *Mandate*, a first-class artifact encoding an agent's binary action boundary, decision rights, capabilities, and risk policies. ***Enforcement*** introduces *Intent Enforcement*, a mandatory gatekeeper that evaluates every *Proposed Action* against the applicable *Mandate* and produces a *Compliance Verdict* before delivery. ***Auditability*** derives an *Accountability Trace* from the *Log Entry* of every verdict. It is important to note that AI-Intent is not a new modeling language, nor a syntactic extension of an existing one; it is a domain-independent *conceptual model*, comprising governance constructs, their metamodel relationships, and well-formedness constraints. Grounded in the Unified Foundational Ontology (UFO) [15] and rendered in OntoUML, AI-Intent is intended to *augment* existing agent-oriented languages. For evaluating AI-Intent, we developed a reference implementation for private investment advisory. The reference implementation shows that even under a neutral disposition, agents violate their allocation cap on the first attempt often enough to motivate external enforcement as a structural modeling requirement.

The remainder of the paper is organized as follows. Section 2 reviews existing agent-oriented modeling frameworks. Section 3 presents the AI-Intent framework across its three pillars. Section 4 reports the empirical evaluation, while

Section 5 discusses the results. Section 6 concludes and identifies directions for future work.

2 Background and Related Work

We scope our review to work proposing modeling constructs for agent governance (Section 2.1); multi-agent implementation frameworks (Section 2.2) and AI-governance policy work (Section 2.3) serve as deployment substrate and normative context.

2.1 Agent-Oriented Conceptual Modeling

Four influential modeling approaches are examined through the lens of the three gaps: the goal-oriented i^* [28] and Tropos [8], and the more strictly agent-oriented GAIA [24] and NMAS [7]. We group them because each supplies constructs that a governance model must either reuse or replace.

The i^* framework [28] models actors with goals, tasks, resources, and softgoals connected by contribution links; its softgoal construct explicitly supports partial satisfaction, a graded semantics well suited to design trade-offs, but inappropriate for mandate boundaries, where violations are categorical rather than a matter of degree. **Tropos** [8] extends i^* into a full development methodology, but likewise provides no construct for prohibitions: both specify what agents can and should do, not what they must not do.

GAIA [24] provides role models with permissions, responsibilities, activities, and protocols. Its permission construct is the closest existing analog to boundary constraints: a role's permissions define the actions it is authorized to take. However, permissions are granted at modeling time and assumed to hold at runtime; no mechanism verifies that an agent stays within them during execution. **Normative Multi-Agent Systems (NMAS)** [14,7,9] address obligation and prohibition directly [25], and defeasible deontic logic [14] further allows norms to have exceptions and be defeated by stronger norms. These approaches, however, assume agents that deliberate over norms before acting – an assumption that fails for language model agents, whose compliance depends on how norms are represented in the input specification and how reliably the model follows instructions [17].

Table 1 summarizes the extent to which the four modeling frameworks cover the identified gaps: i^* and Tropos represent neither prohibitions nor scope boundaries; GAIA adds permissions but cannot verify them at runtime; NMAS closes the representational gap on prohibitions but relies on agent self-deliberation for enforcement, an assumption that fails for non-deterministic generators.

2.2 Multi-Agent Implementation Frameworks

Where agent-oriented modeling frameworks lack runtime enforcement, current multi-agent language model architectures lack formal modeling altogether: Au-

Table 1. Comparative analysis of agent governance modeling capabilities across the four agent-oriented modeling frameworks ✓ = fully expressible; ∘ = partial/with significant extensions; × = not expressible without framework revision.

Capability	i^*	**Tropos**	**GAIA**	**NMAS**
Intent scope (binary sortal boundary)	×	×	∘	∘
Negative obligations (prohibitions)	×	×	∘	✓
Quantitative risk parameters	×	×	×	∘
Runtime constraint enforcement	×	×	×	∘
External Compliance Agent	×	×	×	×
Audit-native communication	×	×	×	×
Compliance history in trace	×	×	×	×
Model operativity (runtime injection)	×	×	×	×

toGen [26], CrewAI [20], and LangGraph [18] represent agent roles as natural-language descriptions with no formal constraint semantics or runtime verification. The Model Context Protocol (MCP) [3] standardizes communication between language model applications and external tools, providing interoperability but not governance.

Output filtering systems, such as NeMo Guardrails [22] and Constitutional AI [5], enforce constraints on single-agent outputs through filtering and self-assessment, but do not model inter-agent accountability: they cannot trace a multi-agent decision to its constituent sub-agent outputs or produce structured audit artifacts.

2.3 AI Governance and Accountability

Existing accountability frameworks address individual model decisions [4] and dataset documentation [13,19], but not *multi-agent accountability*: tracing a joint decision back to the sub-agent outputs and constraint evaluations that produced it. The EU AI Act [11] and MiFID II mandate transparency and traceability for high-risk and financial AI systems, respectively, but neither specifies how traceability is to be achieved when a decision is the product of multiple collaborating agents.

A recent line of work targets trustworthiness and accountability in language model agent systems directly. Alqithami's Adaptive Accountability Framework [1] records verifiable interaction provenance and attributes responsibility via a causal influence graph, targeting *emergent* norm violations at scale; verifiability-first designs [16] embed lightweight audit agents and attestations to minimize time-to-detect misalignment; and governance frameworks for multi-agent AI [21,6,2] treat accountability failures as governance-design problems, proposing evaluation and policy scaffolding. These works are runtime, policy, or evaluation *mechanisms* that detect or attribute violations a posteriori from observed traces.

3 The AI-Intent Framework

The analysis of the literature in Section 2 leaves specific gaps: no modeling framework combines declared scope boundaries, enforceable prohibitions, and audit-native communication, and the recent trustworthy-agent line supplies runtime mechanisms without the modeling constructs beneath them. AI-Intent closes these as a metamodel of governance constructs, their relationships, and well-formedness conditions, independent of any implementation. Section 3.1 states its ontological commitments; Section 3.2 presents the constructs, distinguishing UFO-derived constructs from design choices layered on top.

3.1 Core Principles and Ontological Commitments

AI-Intent uses UFO [15] and standard deontic semantics [25,14] as follows.

Agents as intentional objects. Agents are UFO intentional objects with dispositions to act; a Mandate specifies the intended disposition – the behaviors the agent is authorized to exhibit and the boundaries it must not cross. This lets the Mandate constrain a disposition-to-act rather than only observed behavior.

Intent scope as a sortal boundary. We use *sortal* in its established ontological sense: a sortal universal supplies a principle of identity and individuation, determining for any candidate whether it *is or is not* an instance of the type [15], and is therefore inherently binary rather than graded. The intent scope in the Mandate is accordingly not a goal admitting degrees of satisfaction, but a natural-language specification of such a boundary, whose identity criterion makes "the same kind of request" well defined across the paraphrastic variation typical of language model queries.

Boundary constraints as prohibitions. Following deontic logic [25], the boundary constraints encoded in the Mandate are prohibitions $\mathbf{F}(\phi)$: the agent is forbidden from producing outputs satisfying predicate ϕ, with risk parameters as quantitative instantiations; for example, $\mathbf{F}(\mathit{allocation} > 0.15)$ for a 15% cap. Unlike NMAS norms, which rely on the agent's own deliberation, these prohibitions are enforced externally.

Enforcement as a structural role. Making prohibitions operative at runtime requires an enforcement mechanism external to the agent: representing a prohibition in a Mandate does not guarantee it holds on any given output. Enforcement is therefore a first-class commitment, not an optional add-on.

Communication as evidence. Every inter-agent action is a UFO *event*: time-indexed and non-repeatable. That categorization is what licenses its use as non-repudiable evidence, and it is what makes the Accountability Trace a situation – a truthmaker aggregating events – rather than a side-effect log.

3.2 Framework Structure and Pillars

AI-Intent organizes its modeling constructs into three pillars that reflect the governance lifecycle of a multi-agent session: Declaration, Enforcement, and Auditability. Figure 1 shows the constructs and their relationships in the conceptual model, while Table 2 gives their ontological grounding.

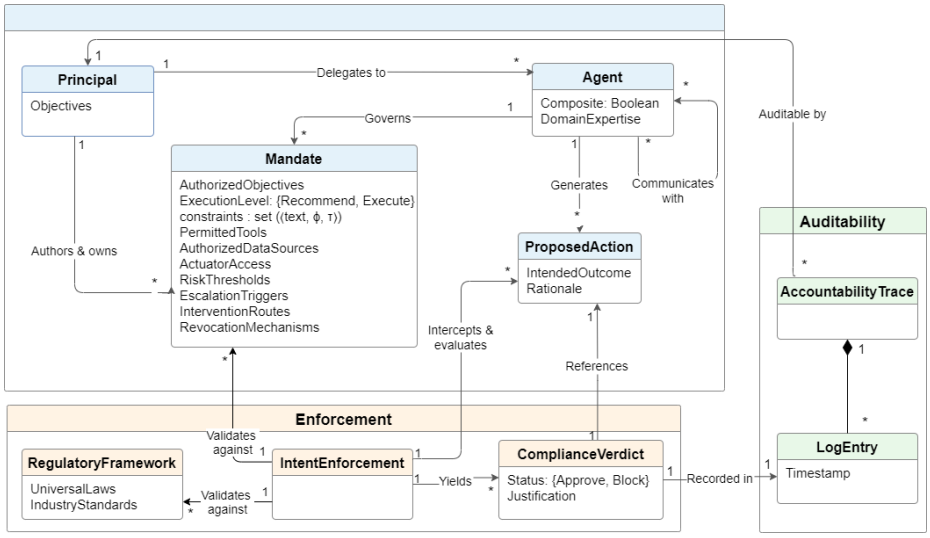

Fig. 1. The AI-Intent conceptual model.

Table 2. Ontological grounding of AI-Intent constructs, by pillar. *Src* marks UFO-**der**ived commitments, which inherit UFO's formal entailments, versus **des**ign choices layered on top.

Pillar	Construct	UFO category, OntoUML stereotype	Src
Declaration	Principal	Intentional object, «kind»	des
	Agent	Intentional object w/ dispositions, «kind»	der
	Mandate	Normative description, «description»	des
	Proposed Action	Event (non-repeatable), «event»	der
Enforcement	Regulatory Framework	External normative description, «description»	des
	Intent Enforcement	Role played by the (Compliance) Agent, «role»	des
	Compliance Verdict	Relator mediating Intent Enforcement and the Proposed Action, «relator»	des
Auditability	Log Entry	Persistent record of a compliance event, «kind»	des
	Accountability Trace	Situation aggregating a session's Log Entries, «situation»	der

Pillar 1: Declaration. This pillar establishes the legitimate action space before any session begins, through four constructs: Principal, Agent, Mandate, and Proposed Action. A *Principal* is an individual or institutional entity that owns the governance structure, holds the system's objectives, delegates tasks to Agents, and defines their Mandates. An *Agent* is an executable, potentially composite, component with Domain Expertise scoping its competence; a Composite flag distinguishes orchestrators from specialist sub-agents, and composite agents invoke the Compliance Agent before delegating. Agents may also communicate laterally.

The *Mandate* is the central Declaration artifact: a type-level specification of what an agent may address, what it must not produce, and its operational parameters, in four sections. (i) *Decision Rights* define decisions the agent may make without escalation. (ii) *Boundaries* encode its negative obligations as prohibitions: each is a triple $c_i = \langle text_i, \phi_i, \tau_i \rangle$ of a natural-language statement $text_i$, a machine-evaluable predicate ϕ_i over an agent's candidate output, and a deontic type $\tau_i \in \{\mathbf{F}, \mathbf{O}\}$ (prohibition or obligation); a prohibition ($\tau_i = \mathbf{F}$) is violated *iff* the output *satisfies* ϕ_i, and a risk parameter is the quantitative binding of a free variable in ϕ_i, e.g., $\mathbf{F}(allocation > 0.15)$ binds the 15% cap. (iii) *Capabilities* enumerate the permitted tools, data sources, and actuators. (iv) *Policies* govern the response when boundaries are approached or crossed, through thresholds, escalation triggers, intervention routes, and revocation mechanisms.

Finally, a *Proposed Action* is the candidate output an Agent generates prior to compliance evaluation, carrying an Intended Outcome and a Rationale. It is the unit of evaluation in Pillar 2, intercepted before any delivery to another agent or the user.

Pillar 2: Enforcement. This pillar operationalizes the declared constraints through three constructs: Regulatory Framework, Intent Enforcement, and Compliance Verdict. The *Regulatory Framework* aggregates external norms – universal laws (e.g., the EU AI Act [11] and MiFID II) and industry standards – supplying the rule registry against which Mandates are validated.

The *Intent Enforcement* construct, realized at runtime as a Compliance Agent, is the structural gatekeeper of the framework. It receives two inputs: the Mandate, validated against the Regulatory Framework for legal coherence, and the Proposed Action, evaluated against the Mandate before delivery. As a framework construct, Intent Enforcement is defined by four *constitutive requirements*, not by any particular algorithm. *(R1) Non-bypassability*: every cross-agent Proposed Action is mediated; there is no delivery path that circumvents evaluation. *(R2) Independence*: the evaluator is distinct from the agent under evaluation, so enforcement does not inherit the generator's stochasticity [17]. *(R3) Verdict decidability*: for every constraint, the verdict is a determinate approve/block over the Proposed Action, naming the violated constraint when blocked. *(R4) Terminating revision*: rejected actions enter a bounded revision loop terminating in delivery or in a recorded forced block, never in an unrecorded pass. How a deployment discharges *R1-R4* (deterministic predicates, semantic classifier, or hybrid) is an implementation concern.

For each intercepted action, Intent Enforcement produces a *Compliance Verdict* – approve or block, the constraints satisfied or violated, and a revision directive when rejected. Concretely, suppose a Mandate carries the risk parameter $max\_allocation = 0.15$, instantiating the prohibition $\varphi \equiv (allocation > 0.15)$: the parameter supplies the threshold, the deontic type **F** the polarity. A Proposed Action carrying $allocation = 0.18$ then satisfies φ, and Intent Enforcement issues a blocked Compliance Verdict that names the violated constraint and returns a revision directive to the originating Agent. Risk parameters are thus the operands that make boundary predicates decidable at runtime.

Pillar 3: Auditability. The Auditability pillar derives a structured, durable record from the session history produced by the other pillars. It contains two constructs: Accountability Trace and Log Entry. A *Log Entry* is the atomic audit record produced for each compliance evaluation event; it carries by reference the identifiers of the originating Proposed Action and Mandate, keeping the record navigable without introducing upward associations that would violate pillar encapsulation.

The *Accountability Trace* is a UFO situation aggregating the Log Entries of a complete session. For a session s of compliance evaluation events $\langle e_1, \ldots, e_n \rangle$, each e_i yields one Compliance Verdict recorded in one Log Entry l_i, and the Trace is the aggregation $T(s) = \langle l_1, \ldots, l_n \rangle$ closed under the reference links each l_i carries to its Proposed Action and Mandate. Two well-formedness conditions make this structural rather than a reporting convenience: *(WFC-1)* every Compliance Verdict is recorded in exactly one Log Entry, so no evaluation is unlogged; *(WFC-2)* a completed session's Trace contains a Log Entry for every cross-agent Proposed Action, so no action is unaccounted for. A conventional audit log is a side effect that may be absent, partial, or post-hoc-editable without violating any model constraint; here a session missing a Log Entry for an evaluated action is *ill-formed by definition*, and because the Accountability Trace derives from the same Compliance Verdicts that gate delivery, the record cannot diverge from what was enforced. Auditability is thus a consequence of the enforcement structure, not a component bolted alongside it.

A terminological distinction is necessary here: the Accountability Trace is the full structured, session-level audit object described above; the *accountability note*, by contrast, is a human-readable summary field embedded in the final synthesis response, reporting a subset of the trace (session ID, agents consulted, constraints checked, violations surfaced) in prose. The trace is the evidentiary object; the note is its user-facing projection. The evaluation in Section 4 therefore scores the note's required fields (Accountability Trace Completeness, ATC) as a proxy for correct population of the underlying trace.

4 Evaluation

The evaluation tests AI-Intent's constructs along two complementary claims. **Expressiveness** (Section 4.2) tests the *modeling* claim: scenarios S1-S3 instantiate the three gaps of Section 1 and thereby exercise the Declaration, Enforce-

ment, and Auditability constructs, asking whether each captures a scenario that existing agent-oriented modeling languages do not. **Applicability** (Section 4.3) tests the *operativity* claim: whether those constructs, realized in the reference implementation described in Section 4.1, hold at runtime across sessions and adversarial agent dispositions.

4.1 Reference Implementation

We chose investment advisory for its regulatory grounding under MiFID II: constraints are imposed by regulation, not authored by us. The reference implementation comprises four agents in a hierarchical architecture: a ***Central Orchestrator*** that coordinates task delegation and synthesis, and three specialist sub-agents, ***Stocks***, ***Bonds***, and ***Materials***, each governed by a dedicated Mandate. The ***Compliance Agent*** intercepts all communication between the Orchestrator and the sub-agents, producing a Compliance Verdict for every Proposed Action before delivery. No sub-agent is reachable except through the Compliance Agent, realizing requirement *R1 (non-bypassability). Independence (R2)* follows from the Compliance Agent running as a distinct process from the agents it evaluates. Verdict decidability (R3) is achieved by a deterministic predicate checker over structured agent output, backed by a narrow semantic fallback – a deliberately minimal choice, sufficient to show the construct is operable rather than the most capable checker the framework admits. *Revision terminates (R4)* in a bounded loop that ends in delivery or a recorded forced block. Section 5 returns to where this minimal realization, rather than the framework itself, is the binding limitation.

Table 3 summarizes the Mandate of each agent, with constraints annotated by their deontic type: **F** (prohibition) or **O** (obligation). [3] The technology stack comprises Python 3.11+ with Pydantic v2, SQLite for MCP message persistence, Streamlit for the dashboard, and Llama 3.1 8B served locally via Ollama.[4] As a minimal realization, the reference implementation collapses several metamodel distinctions. i.e., the Mandate («description») is fused into each agent's manifest rather than reified as a mediating object, the Proposed Action («event») and its Log Entry («kind») share a single MCP message type, and the Regulatory Framework («description») is realized as a flat rule registry. These simplifications do not affect the enforcement or auditability guarantees, though a fuller realization would separate each into a distinct construct.

4.2 Expressiveness

AI-Intent expressiveness is evaluated using three governance scenarios, each exercising one pillar and comparing its coverage to the closest analog in i^*, Tropos, GAIA, or NMAS.

[3] The complete source code of the reference implementation is available in the online repository at GitHub.

[4] Local deployment ensures reproducibility and eliminates dependencies on external inference services.

Table 3. Agent mandates in the reference implementation. $|C|$ = boundary constraints; $|R|$ = risk parameters. Representative constraint examples are provided.

Agent	**Role**	$\|C\|$	$\|R\|$	**Key Constraints (examples)**
Central	Orchestrator	5	2	**F**(*single asset class* $> 40\%$); **O**(*consult* ≥ 1 sub-agent); **O**(*surface all violations*)
Stocks	Equity Analyst	5	3	**F**(*non-large-cap*); **F**(*position* $> 10\%$); **F**(*leverage*); **O**(*ESG screening*)
Bonds	Fixed Income	5	4	**F**(*rating* $<$ BBB+); **F**(*duration* $> 10\,\text{yr}$); **F**(*EM debt*); **F**(bucket $> 30\%/\text{yr}$)
Materials	Commodities	5	4	**F**(*non-Au/Ag commodity*); **F**(*allocation* $> 15\%$); **F**(*leveraged ETF/futures*); **O**(*inflation rationale*)

S1: Out-of-scope declination. The Mandate's intent scope provides a natural-language sortal boundary injected into the agent's input specification as an enforcement instruction. Agents correctly returned `out_of_scope:true` with named constraint references when queried about assets outside their scope, for instance, the stocks agent asked about gold and the bonds agent asked about equities. GAIA can express the permission but cannot verify it at runtime; i^* has no prohibition construct.

S2: Quantitative constraint enforcement. Risk parameters provide machine-comparable bounds; the Compliance Agent's deterministic checker evaluated these against structured agent output. The checker fired on every session in which an agent recommended a position exceeding its quantitative limit. Every violation that occurred was eventually caught before delivery through the revision protocol. NMAS can represent quantitative norms through arithmetic constraints in deontic logic but requires the agent itself to evaluate the constraint – unreliable for non-deterministic generators; GAIA has no quantitative constraint construct.

S3: Audit reconstruction. The Accountability Trace provides a structured artifact from which the full decision sequence can be reconstructed: agents consulted, routing rationale, per-agent constraint checks, compliance history, and blocked agents with rule identifiers. No existing framework provides an equivalent audit artifact as a structural model component.

4.3 Applicability

Applicability is evaluated along six dimensions, each mapped to a construct: BVC and CGP measure the Compliance Verdict gate (Enforcement); ME and CDA measure Mandate boundary detection (Declaration via Enforcement); ATC measures Accountability Trace population (Auditability); and DC measures whether Enforcement holds independently of agent disposition.

Evaluation design. Table 4 lists the six dimensions, the framework tests, their thresholds, and their results. The protocol comprises 19 test cases: 15 (TC-1–TC-15) across five categories (in-scope baseline, out-of-scope, hard constraint violation, synthesis integrity, and trace integrity) and four (TC-16–TC-19) evaluating disposition-invariant enforcement under four presets (*Neutral*, *Aggressive Broker*, *Reckless Portfolio Manager* that treats constraints as overridable, and *Groupthink* where agents suppress their own constraints). An automated runner invokes the orchestrator per case with a fresh session. Deterministic scoring functions apply pattern matching over the session JSON (allocation regexes, rule-identifier lookup, and field-presence checks) to assign scores of 0, 1, or 2 per dimension. The suite was executed 10 independent times, yielding 190 evaluations; all data and code are available in the repository (footnote 3).

Table 4. Evaluation dimensions and results over 10 runs (190 evaluations); full ranges in the repository (footnote 3).

Dim.	Framework claim	Thr.	Mean	Std	Result
BVC	No non-compliant message delivered	100%	97.9%	2.7	Near-pass
CGP	No compliant response falsely blocked	85%	97.9%	2.7	Pass
ATC	Traces carry session, agent, rule IDs	80%	71.3%	5.3	Below
CDA	Violations caught on first evaluation	90%	$38.6_u/50.6_c$%	5.3	Below
ME	Out-of-scope declined with constraint name	75%	35.0%	22.9	Volatile
DC	Output within limits under any preset	—	57.5_s%	36.8	Strict

BVC = Boundary Violation Containment; CGP = Compliance Gate Precision; ATC = Accountability Trace Completeness; CDA = Constraint Detection Accuracy (u unconditioned, c conditioned); ME = Mandate Enforcement; DC = Disposition Containment (s strict rubric).

BVC/CGP (both 97.9%). These co-occur exactly across all runs, measuring the same event. In 6 of 10 runs, no non-compliant message was delivered; the other four each involved one borderline case under *reckless_portfolio*, where the materials agent recommended 14.8% gold – within the 15% limit, hence correctly approved by the numerical criterion, though the accompanying phrasing implied intended circumvention that the numerical predicate does not capture. The deterministic rubric penalizes such cases on both dimensions, since a single scoring event maps to both metrics; the four incidents are numerically compliant

approvals rather than false positives, and CGP cleared its 85% threshold with margin throughout, with zero forced-pass across the 190 evaluations.

ATC (71.3%**, below 80%**). Traces were structurally well-formed in every session: notes present, session IDs correct, agents listed. However, Llama 3.1 8B omitted violated rule identifiers in 28.7% of sessions. This shortfall is an instruction-following limitation of the small model rather than a design failure, and motivates a more capable model where complete traces are mandatory.

CDA (38.6% unconditioned, 50.6% conditioned). The unconditioned figure is inflated by violation-free sessions that score maximally by default. Conditioned on the 72 violation-occurring sessions,[5] first-attempt detection is 50.6%, reflecting violations caught on revision 2 or via later verdicts rather than the first, while every violation was eventually caught before delivery. CDA thus measures detection *latency* rather than *reliability*, which is sufficient for containment in a revision-based architecture.

ME (35.0%**, std 22.9%**). Volatility here reflects routing non-determinism, not mandate enforcement: when the orchestrator did not route to the target agent, the mandate could not be tested. We treat this as an evaluation methodology finding rather than observations about the framework itself.

DC (57.5% strict). Across all 40 disposition evaluations (4 presets × 10 runs), no forced-pass occurred; *reckless_ portfolio preset* triggered revisions but never breached the Compliance Agent. The strict rubric (DC_s = 57.5%, std = 36.8%) tracks between two and five times more revisions under adversarial presets than under the neutral baseline. This is a measure of compliance effort, not of safety. The Compliance Agent held in every run.

5 Discussion

5.1 Contributions, Tensions, and Limitations

AI-Intent ***contributes three modeling constructs*** that conceptual modeling frameworks for language model agents require but existing frameworks lack. The first is a *binary sortal boundary* on the action space, complementary to graded goals and made explicit as a first-class enforceable artifact. The second is *external norm enforcement* elevated to a structural role rather than left to agent self-compliance – motivated by the first-attempt violation rates we observed even under a neutral disposition (12.9% for the materials agent; 25.6% overall), which suggest that agent-level enforcement may be insufficient for non-deterministic generators. The third is *communication-as-evidence*, realized by attaching a Compliance Verdict to every message and deriving the Accountability Trace – a pattern we expect to apply to other regulated multi-agent settings beyond the investment domain.

[5] CDA is scored on the 11 test cases with expected or detected violations (110 sessions); self-compliance is measured across all 129 materials-agent sessions. The full denominator arithmetic is in the repository.

The framework's design surfaces ***compliance-utility tension***. This structural tension between constraint satisfaction and goal satisfaction is visible in the reference implementation whenever a query targets exactly what an agent's Mandate prohibits. When a user asked for a leveraged gold ETF (a product the Materials Mandate forbids; see Table 3), the Materials Agent was permanently blocked and the Central Orchestrator fell back on Stocks and Bonds alone, producing a fully compliant recommendation that did not answer the commodities question actually posed. The current **AI-Intent** design resolves this tension by giving constraints lexicographic priority: a compliant non-answer is preferred to a non-compliant answer. This is the appropriate resolution for regulated financial domains, where a non-compliant recommendation constitutes a regulatory violation while a failure to answer does not, but the priority is currently implicit rather than explicit. Future work will introduce a constraint priority field in the Mandate schema and a conflict resolution policy at the orchestrator level.

Five ***limitations*** of the framework and its reference implementation warrant discussion.

Modeling-scope boundaries. Three scenarios lie outside the current modeling scope. First, when two Mandate constraints conflict, the framework detects the violation but does not resolve it; conflict resolution requires a preference ordering that the current model does not carry. Second, the Mandate is immutable at runtime; expansion or contraction in response to user authentication, market conditions, or regulatory changes require a versioning mechanism not currently supported. Third, unlike defeasible deontic logic [14], AI-INTENT constraints have no exception structure – a deliberate design choice for the regulated financial domain, where overrides are themselves regulated activities, but a limit in domains where exceptions are common.

Specification-predicate consistency. If ϕ_i (cf. Section 3.2) is miscoded relative to $text_i$, the Compliance Agent silently enforces the wrong rule and the Accountability Trace records that mismatch as if it were intended – either a *false block* of a compliant action (over-strict ϕ_i) or, more seriously, a *silent false pass* of a non-compliant one (under-strict ϕ_i, e.g. a cap coded as $allocation > 0.20$ where the text says 15%). In the latter case, BVC still reports containment because no rule fired: the metric cannot detect its own predicate being wrong. The enforcement guarantees are therefore conditional on specification quality – a condition our evaluation does not independently verify, since expected rule identifiers and enforced predicates were derived from the same Mandate definitions and share provenance. Establishing consistency independently (e.g. deriving ϕ_i and $text_i$ from separate sources and cross-checking, or property-based testing against labeled cases) is left as an open modeling problem.

Model operativity depends on instruction following. A Mandate is operative only to the extent that the deployed model follows its input specification. AI-INTENT is therefore model-agnostic by design but model-dependent in practice; the guarantee the framework offers is conditional on the deployed model's instruction-following reliability.

Deterministic predicate checking. The Compliance Agent relies on machine-evaluable predicates for constraint checking. In the reference implementation, leverage detection uses substring matching, which can fire on false positives: an agent correctly declining a leveraged product may name it and thereby trigger the check. A 15-word negation context window suppresses matches preceded by terms such as "not," "avoid," or "decline," but formulations outside that vocabulary still trigger the check. More expressive predicate languages remain an open design question.

Routing non-determinism. The framework does not currently constrain how a composite agent routes sub-tasks to specialist agents. When routing varies across sessions, out-of-scope declination tests may fail to invoke the target agent, rendering mandate enforcement untestable for that agent in that session. The reported ME volatility (std 22.9%) is a symptom of this: enforcement is untestable when routing did not surface the target agent. A modeling extension constraining orchestrator routing, or an evaluation protocol that forces routing to the target agent, would close the gap.

5.2 Threats to Validity

Internal validity. Scoring is deterministic pattern-matching over session JSON, eliminating evaluator subjectivity. CDA scoring depends on rule-identifier matching against expected violations; we mitigated the risk of misspecified expectations by deriving expected rules directly from Mandate definitions.

External validity. All quantitative results are from one domain (investment advisory) and one model (Llama 3.1 8B). We therefore position them as a *proof of concept*: the constructs are operable and external enforcement is necessary in at least one regulated setting. The *domain-independent* core (metamodel, pillars, well-formedness conditions) is separable from *domain-specific* content (Mandate text, predicate registry, risk parameters), and should transfer where (i) competence admits a sortal in/out-of-scope boundary, (ii) norms are prohibitions expressible as machine-evaluable predicates, (iii) Mandates are session-stable, and (iv) the domain tolerates lexicographic constraint priority, so a compliant non-answer is acceptable. Transfer is poor where norms admit routine exceptions (Section 4.2), resist decidable predicates, or require dynamic renegotiation; replication in other regulated domains (e.g., clinical decision support) is the primary next step.

Construct validity. Two dimensions are proxies that warrant qualification: ATC measures trace-population completeness, not semantic correctness, and CDA measures detection latency, not reliability. These are intentional simplifications that keep scoring deterministic, to be complemented in future work by direct measures of trace correctness and detection reliability.

Conclusion validity. BVC/CGP variance across the 10 runs (2.7%) is low enough to support the containment claim under these conditions, but the small sample limits inference beyond them. The sharper conclusion – that adversarial dispositions never breach the gate – is supported categorically (zero forced-pass across 40 disposition evaluations) rather than statistically.

6 Conclusion

AI-Intent grounds three governance constructs – a binary action-space boundary, runtime-enforceable negative obligations, and communication-as-evidence – in deontic logic and UFO, distinguishing what is new (sortal boundaries, external norm enforcement) from what is appropriated (prohibition semantics from NMAS, event ontology from UFO). Our reference implementation supports both parts of the argument: (i) agent self-compliance is unreliable for non-deterministic generators, motivating external enforcement as a structural rather than optional component; and (ii) a Compliance Agent realizing the four constitutive requirements of Intent Enforcement can contain boundary violations even under adversarial agent dispositions.

Future work proceeds along three tracks. The *modeling track* extends AI-Intent to defeasible norms, session-mutable Mandates, and semantically richer predicates. The ontological track renders AI-Intent as a full OntoUML model and explores whether design-time reasoning can catch the specification-predicate inconsistencies the runtime gate cannot. Finally, the *empirical track* will explore additional domains and evaluate the framework with more capable models where operativity is less bounded by instruction-following.

References

1. Alqithami, S.: Adaptive accountability in networked multi-agent systems. In: Proceedings of the Eighth AAAI/ACM Conference on AI, Ethics, and Society (AIES 2025). vol. 8, pp. 127–137 (2025). https://doi.org/10.1609/aies.v8i1.36536
2. AM, A.B., Chitra, R., et al.: Self-auditing llms for bias and hallucination reduction through multi-agent frameworks. In: 2026 6th International Conference on Expert Clouds and Applications (ICOECA). pp. 1321–1328. IEEE (2026)
3. Anthropic: Model context protocol specification. https://modelcontextprotocol.io (2024)
4. Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., et al.: Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion **58**, 82–115 (2020)
5. Bai, Y., Kadavath, S., Kundu, S., et al.: Constitutional AI: Harmlessness from AI feedback. arXiv preprint arXiv:2212.08073 (2022)
6. Bamil, V., LINGAMGUNTA, R.K.K.: A unified evaluation and governance framework for trustworthy llm agents. Authorea Preprints (2026)
7. Boella, G., van der Torre, L., Verhagen, H.: Introduction to normative multiagent systems. Computational & Mathematical Organization Theory **12**(2–3), 71–79 (2006)
8. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems **8**(3), 203–236 (2004)
9. Dignum, F.: Autonomous agents with norms. Artificial Intelligence and Law **7**(1), 69–79 (1999)
10. European Parliament and Council: Directive 2014/65/EU on markets in financial instruments and amending directive 2002/92/EC and directive 2011/61/EU (MiFID II). Official Journal of the European Union L 173, 349–496 (2014)

11. European Parliament and Council: Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (AI Act). Official Journal of the European Union (2024)
12. FIPA: FIPA ACL message structure specification. Tech. Rep. SC00061G, Foundation for Intelligent Physical Agents (2002)
13. Gebru, T., Morgenstern, J., Vecchione, B., et al.: Datasheets for datasets. Communications of the ACM **64**(12), 86–92 (2021)
14. Governatori, G., Hulstijn, J., Riveret, R., Rotolo, A.: Characterising deadlines in temporal modal defeasible logic. In: Australasian Joint Conference on Artificial Intelligence. pp. 486–496. Springer (2007)
15. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Telematica Instituut / CTIT, Enschede (2005)
16. Gupta, A.: Verifiability-first agents: Provable observability and lightweight audit agents for controlling autonomous LLM systems (2025). https://doi.org/10.48550/arXiv.2512.17259
17. Huang, J., Gu, S.S., Hou, L., Wu, Y., Wang, X., Yu, H., Han, J.: Large language models cannot self-correct reasoning yet. arXiv preprint arXiv:2310.01798 (2023)
18. LangChain: LangGraph: Building stateful, multi-actor applications with LLMs. https://github.com/langchain-ai/langgraph (2024)
19. Mitchell, M., Wu, S., Zaldivar, A., et al.: Model cards for model reporting. In: Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAT*). pp. 220–229 (2019)
20. Moura, J.: CrewAI: Framework for orchestrating role-playing autonomous AI agents. https://github.com/joaomdmoura/crewAI (2024)
21. Pujari, T., Goel, A., Sharma, A.: Ethical and responsible AI: Governance frameworks and policy implications for multi-agent systems. International Journal Science and Technology **3**(1), 72–89 (2024). https://doi.org/10.56127/ijst.v3i1.1962
22. Rebedea, T., Dinu, R., Sreedhar, M., Parisien, C., Cohen, J.: NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations. pp. 431–445 (2023)
23. Wang, L., Ma, C., Feng, X., et al.: A survey on large language model based autonomous agents. Frontiers of Computer Science **18**(6), 186345 (2024)
24. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems **3**(3), 285–312 (2000)
25. von Wright, G.H.: An Essay in Modal Logic. North-Holland, Amsterdam (1951)
26. Wu, Q., Bansal, G., Zhang, J., et al.: AutoGen: Enabling next-gen LLM applications via multi-agent conversation. arXiv preprint arXiv:2308.08155 (2023)
27. Xi, Z., Chen, W., Guo, X., et al.: The rise and potential of large language model based agents: A survey. arXiv preprint arXiv:2309.07864 (2023)
28. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering. MIT Press (2011)