RAGF: Bridging Probabilistic AI Reasoning and Deterministic Execution in Regulated Systems

A Governance Framework for Certifiable Agentic AI

Yamil Rodríguez Montaña
RefleXio
Barcelona, Spain
yrm@reflexio.es

Abstract

The deployment of Large Language Model (LLM)-based agentic systems in regulated industries—aviation, healthcare, defense, and critical infrastructure—faces a fundamental challenge: **how to bridge probabilistic AI reasoning with the deterministic execution required for safety certification**. Current approaches either constrain AI capabilities to predetermined playbooks or accept unverifiable "black box" autonomy, neither of which satisfies regulatory frameworks like DO-178C, ISO 42001, or the EU AI Act.

We present the **Reflexio Agentic Governance Framework (RAGF)**, a novel architecture that enables certifiable agentic AI through three core innovations: (1) *Semantic Governance*—domain ontologies that constrain action spaces to formally modeled operations, (2) *The Validation Gate*—a deterministic, independent verification layer that enforces safety invariants regardless of AI reasoning quality, and (3) *Cryptographic Non-Repudiation*—immutable audit trails with HMAC-SHA256 signatures for compliance traceability.

We demonstrate RAGF's viability through implementation in aviation route optimization (AMM Level 3) and healthcare clinical decision support (AMM Level 2-3), achieving <30ms p95 validation latency while maintaining 100% fail-closed guarantees under all tested failure modes. Our empirical validation of the safety property $\forall \text{failure} \in \text{FailureModes} : \text{evaluate}(\text{action}) = \text{DENY}$ aligns with DO-178C Level C verification objectives for deterministic behavior under failure conditions.

Key Contribution: RAGF introduces a *commitment boundary* that separates "thinking" (probabilistic, uncertifiable) from "doing" (deterministic, certifiable), enabling organizations to deploy agentic AI in regulated contexts without certifying the LLM itself—only the governance harness.

CCS Concepts

• **Computer systems organization** → **Embedded systems**; • **Computing methodologies** → **Artificial intelligence**; • **Security and privacy** → *Software security engineering*.

Keywords

Agentic AI, LLM Safety, Governance, Certification, DO-178C, Semantic Validation, Safety-Critical Systems

1 Introduction

The emergence of Large Language Models (LLMs) with reasoning capabilities—demonstrated by systems like GPT-4, Claude, and Gemini—has created unprecedented opportunities for autonomous decision-making in complex domains. However, the deployment of these *agentic AI systems* in regulated industries faces a critical barrier: **regulatory frameworks require deterministic, auditable, and certifiable behavior**, while LLMs are inherently probabilistic, opaque, and non-deterministic.

1.1 The Certification Gap

Consider a concrete example from aviation: an AI agent proposes rerouting Flight IB3202 to optimize fuel consumption, extending crew duty time by 45 minutes. A traditional LLM might generate this action based on pattern matching across millions of flight records, achieving 95% confidence in fuel savings. However, this violates FAA 14 CFR §91.1057 (crew rest requirements) by 15 minutes—a regulation the model may not explicitly encode in its reasoning chain.

Current approaches to this problem fall into two inadequate categories:

(1) **Constrained Playbooks**: Limit AI to predefined decision trees, sacrificing the adaptive intelligence that makes LLMs valuable.
(2) **Human-in-the-Loop (HITL)**: Require human approval for every action, creating bottlenecks that negate automation benefits.

Neither approach satisfies the core regulatory requirement: *demonstrable safety under all failure modes*, including AI hallucination, data drift, adversarial inputs, and system outages.

1.2 The RAGF Insight

RAGF's foundational insight is that **we don't need to certify the AI—we need to certify the governance layer**. By separating the *reasoning engine* (probabilistic, adaptive, uncertifiable) from the *validation gate* (deterministic, verifiable, certifiable), we create a system where:

- The AI can propose any action within its capabilities
- A formal verification layer ensures *only semantically valid and regulation-compliant actions execute*
- All rejections are logged with cryptographic signatures for audit trails
- The system fails closed: **any error in validation results in DENY**

This architecture mirrors the principle used in safety-critical hardware: a watchdog timer that operates independently of the main processor. RAGF is the "watchdog" for agentic AI.

1.3 Contributions

This paper makes the following contributions:

(1) **The Validation Gate Architecture**: A formally specified, independent validation layer that enforces semantic and regulatory constraints on AI-proposed actions (Section 3).
(2) **Semantic Governance via Domain Ontologies**: A methodology for encoding business rules, safety regulations, and domain knowledge as machine-readable ontologies that constrain AI action spaces (Section 4).
(3) **Cryptographic Non-Repudiation**: HMAC-SHA256 signatures on all validation decisions, creating immutable audit trails for compliance (Section 5.4).
(4) **Empirical Fail-Closed Property**: Validation that $\forall$failure : evaluate(action) = DENY, aligning with DO-178C Level C verification objectives for deterministic failure behavior (Section 6).
(5) **Production Implementation**: Deployment in aviation and healthcare domains, demonstrating <30ms p95 latency and 100% safety coverage (Section 7).
(6) **Agentic Maturity Model (AMM)**: A 5-level taxonomy for classifying AI autonomy and mapping to governance requirements (Section 2.2).

1.4 Paper Organization

Section 2 provides background on agentic AI and regulatory challenges. Section 3 presents the RAGF architecture. Sections 4-5 detail the Semantic Layer and Validation Gate implementations. Section 6 empirically validates the fail-closed safety property through comprehensive testing. Section 7 evaluates production deployments. Section 8 discusses limitations and future work.

2 Background and Motivation

2.1 The Shift from Conversation to Execution

Traditional LLM applications terminate with text generation: a chatbot provides an answer, a writing assistant suggests edits. These are *interfaces*, not *commitments*. The risk of hallucination is misinformation, not operational failure.

Agentic AI crosses the execution boundary: the system initiates actions in the real world—rerouting flights, adjusting medical treatments, controlling industrial processes. This shift introduces three critical challenges:

(1) **Regulatory Frameworks**: Systems become subject to DO-178C (aviation), IEC 61508 (industrial), IEC 62304 (medical devices), and ISO 42001 (AI management).
(2) **Safety Standards**: Actions must satisfy deterministic safety constraints, not probabilistic confidence thresholds.
(3) **Operational Liability**: Organizations bear legal responsibility for AI-initiated actions, requiring demonstrable due diligence.

2.2 The Agentic Maturity Model (AMM)

We propose a 5-level taxonomy for classifying AI system autonomy:

The L2→L3 Transition is the Critical Inflection Point: At L2, humans act as the validation gate. At L3, the system must provide an *independent, deterministic validator* to replace human judgment. RAGF is designed specifically for this transition.

Table 1: Agentic Maturity Model (AMM)

Level	Name	Capability
L1	Passive Knowledge	Read-only queries, document retrieval
L2	Human Teaming	AI proposes, human executes
L3	Actionable Agency	AI executes with validation
L4	Autonomous Orchestration	Multi-agent coordination
L5	Systemic Autonomy	Self-regulation and adaptation

2.3 Regulatory Landscape

Three regulatory frameworks drive RAGF's design:

2.3.1 DO-178C (Aviation Software). Requires demonstrable safety under failure conditions (§11.10, paraphrased):

> *Software shall not allow hazardous operations under failure conditions, including component failures, invalid data, and loss of redundancy.*

RAGF Alignment: The Validation Gate's fail-closed property contributes to this objective by ensuring **DENY** on any validation failure.

2.3.2 EU AI Act (Article 12). Mandates transparency and traceability for high-risk AI (paraphrased):

> *High-risk AI systems shall be designed to ensure an adequate level of traceability through recording of events.*

RAGF Alignment: Cryptographic signatures create tamper-evident audit logs that support this traceability requirement.

2.3.3 ISO 42001 (AI Management). Requires risk management and validation mechanisms for AI systems.

RAGF Alignment: Semantic ontologies combined with validation gates provide technical mechanisms that support ISO 42001 risk control requirements.

2.4 Why Existing Approaches Fail

2.4.1 Guardrails (Prompt Engineering). Systems like Anthropic's Constitutional AI or OpenAI's System Messages attempt to constrain model behavior through prompts. **Failure Mode**: Prompt injection, jailbreaking, and semantic drift can bypass text-based constraints.

2.4.2 RLHF (Reinforcement Learning from Human Feedback). Fine-tuning models to align with human preferences. **Failure Mode**: Cannot provide formal guarantees; aligned behavior emerges statistically but isn't provable.

2.4.3 Retrieval-Augmented Generation (RAG). Grounding LLM responses in authoritative documents. **Failure Mode**: The model still interprets retrieved content probabilistically; hallucination remains possible.

RAGF's Distinction: We don't attempt to make the LLM safe—we make the *system* safe by validating *actions*, not *reasoning*.

3 The RAGF Architecture

3.1 Core Principle: Certify the Harness, Not the Core

RAGF's design philosophy: **the LLM is the "thinking engine," the Validation Gate is the "safety interlock."** We achieve certification by:

(1) Accepting that LLMs are non-deterministic and uncertifiable
(2) Building a *deterministic validation layer* that operates independently
(3) Proving formal properties of the validation layer
(4) Auditing the validation logic (small, bounded code) instead of the LLM (billions of parameters)

3.2 Four-Layer Architecture

```
Layer 4: Domain Ontologies
(SNOMED-CT, IEC 61850, FAA 14 CFR)



Layer 3: Business & Safety Rules
(IF confidence < 0.95 THEN escalate)



Layer 2: Governance Ops (CI/CD)
(Automated validation deployment)



Layer 1: State Representation
(TimescaleDB - immutable audit log)
```

Figure 1: RAGF Four-Layer Architecture

3.2.1 *Layer 1: Operational State Representation.*

TimescaleDB stores all validation decisions as immutable time-series events. Every verdict includes:

- Action primitive (verb, resource, parameters)
- Validation decision (ALLOW/DENY/ESCALATE)
- Semantic coverage score (0-1)
- Validator results (PASS/FAIL/TIMEOUT)
- HMAC-SHA256 signature

3.2.2 *Layer 2: Governance Operations.*

CI/CD pipeline for deploying validation logic:

- Version-controlled rule engines
- Automated testing before production
- Blue-green deployment for zero-downtime updates

3.2.3 *Layer 3: Business & Safety Rules.*

Machine-executable constraints:

```
IF action.verb == "prescribe_medication"
   AND action.params["drug"] in CONTRAINDICATED
   AND patient.allergies.contains(drug)
THEN DENY with reason="ALLERGY_CONTRAINDICATION"
```

Listing 1: Example Safety Rule

3.2.4 *Layer 4: Domain Ontologies.*

Formal knowledge graphs (Neo4j) encoding:

- Valid verbs for each domain
- Semantic relationships between concepts
- AMM level authorization per verb

3.3 The Validation Gate: Core Component

The Validation Gate is the critical component that enforces deterministic safety. Its operation follows this sequence:

Algorithm 1 Validation Gate Execution Flow

```
Input: $action$ (ActionPrimitive), $amm\_level$ (1-5)
Output: $verdict$ (ALLOW/DENY/ESCALATE)

$start\_time \leftarrow$ current_time()
if health_check() = FAIL then
    return DENY("VALIDATOR_UNHEALTHY")
end if

$semantic\_verdict \leftarrow$ validate_semantic_authority($action$, $amm\_level$)
if $semantic\_verdict$ = DENY then
    return DENY($semantic\_verdict.reason$)
end if

$validators \leftarrow$ get_required_validators($action$)
if $validators = \emptyset$ then
    return ALLOW("No validators required")
end if

$results \leftarrow$ execute_validators_parallel($validators$, $action$)
if $\exists r \in results : r$ = FAIL then
    return DENY("Validator failed: " + $r.reason$)
end if

$verdict \leftarrow$ aggregate_decisions($semantic\_verdict$, $results$)
$verdict.signature \leftarrow$ HMAC-SHA256($verdict$)
return $verdict$
```

Key Properties:

(1) **Independence**: Validators operate in isolated microservices
(2) **Determinism**: Same input always produces same output
(3) **Fail-Closed**: Any error results in DENY
(4) **Bounded Latency**: 200ms timeout enforced

4 Semantic Governance via Domain Ontologies

Traditional data governance manages *structured data* (databases, files). Semantic governance manages *meaning*—ensuring AI interpretations align with business intent and regulatory constraints.

4.1 The Semantic Alignment Problem

LLMs excel at generating plausible text but lack grounded understanding. Example:

> **User Intent**: "Reduce operating costs for Flight IB3202"
> **LLM Interpretation**: "Reroute via shorter path, extend crew duty time"
> **Regulatory Constraint**: FAA 14 CFR §91.1057 (missed by LLM)

The LLM's reasoning is semantically coherent but *operationally invalid*. RAGF's Semantic Layer prevents execution by encoding regulations as graph constraints.

4.2 Ontology Design Principles

RAGF ontologies follow three principles:

4.2.1 1. Formal Verb Taxonomies. Every action must map to a formally defined verb:

```
CREATE (v:Verb {name: "reroute_flight"})
CREATE (d:Domain {name: "aviation"})
CREATE (r:Regulation {code: "FAA-14-CFR-91.1057"})
CREATE (v)-[:BELONGS_TO]->(d)
CREATE (v)-[:MUST_SATISFY]->(r)
```

Listing 2: Neo4j Ontology Example

4.2.2 2. Semantic Coverage Metric. The ontology query returns a coverage score:

$$\text{coverage} = \frac{\text{matched\_constraints}}{\text{total\_applicable\_constraints}}$$

Coverage < 1.0 triggers human escalation (HITL).

4.2.3 3. AMM-Level Authorization. Verbs are tagged with minimum AMM level:

```
MATCH (v:Verb {name: "prescribe_medication"})
SET v.min_amm_level = 3
```

Agent at Level 2 attempting this verb $\longrightarrow$ DENY.

4.3 Neutralizing Operational Hallucinations

Operational Hallucination: LLM generates a verb outside the ontology.

Example: Agent proposes emergency_landing_override—a verb not in the aviation ontology because it requires manual pilot control.

RAGF Response:

```
semantic_verdict = {
  "decision": "DENY",
  "reason": "Verb 'emergency_landing_override' not
       in ontology",
  "ontology_match": false,
  "coverage": 0.0
}
```

This prevents the system from executing *imagined* capabilities.

4.4 Ontology Maintenance

Ontologies are version-controlled:

- Changes reviewed by domain experts
- Deployed via CI/CD pipeline
- Backward compatibility ensured through semantic versioning
- Deprecated verbs marked but not removed (audit trail preservation)

5 The Validation Gate: Implementation

5.1 Independent Validator Pattern

Each validator is an isolated microservice satisfying three constraints:

(1) **No LLM Calls**: Validators use deterministic logic only
(2) **No Inter-Validator Communication**: Prevents cascading failures
(3) **Bounded Execution Time**: 150ms timeout per validator

```
class BaseValidator(ABC):
    @abstractmethod
    async def validate(
        self, action: ActionPrimitive
    ) -> ValidatorResult:
        """
␣␣␣␣␣␣␣␣Returns:␣ValidatorResult(
␣␣␣␣␣␣␣␣␣␣␣␣decision:␣PASS␣|␣FAIL␣|␣TIMEOUT,
␣␣␣␣␣␣␣␣␣␣␣␣reason:␣str,
␣␣␣␣␣␣␣␣␣␣␣␣latency_ms:␣float,
␣␣␣␣␣␣␣␣␣␣␣␣rule_violated:␣Optional[str]
␣␣␣␣␣␣␣␣)
␣␣␣␣␣␣␣␣"""
        pass
```

Listing 3: Base Validator Interface

5.2 Validator Examples

5.2.1 Aviation: Fuel Reserve Validator. Enforces FAA 14 CFR §91.151:

```
class FuelReserveValidator(BaseValidator):
    async def validate(self, action):
        if action.verb != "reroute_flight":
            return PASS

        route = action.parameters["new_route"]
        fuel_reserve = calculate_reserve(route)

        if fuel_reserve < MINIMUM_RESERVE:
            return FAIL(
                reason="Fuel␣reserve␣below␣FAA␣
                    minimum",
                rule_violated="FAA-14-CFR-91.151"
            )
        return PASS
```

5.2.2 *Healthcare: Drug Interaction Validator.* Validates medication contraindications by checking the prescribed drug against the patient's current medications and known interaction database. Returns `FAIL` if contraindications exist, `PASS` otherwise. Implementation follows the same pattern as the aviation validator (deterministic logic, <150ms execution, no LLM calls).

5.3 Parallel Execution and Aggregation

Validators execute in parallel using `asyncio.gather()`:

```python
results = await asyncio.gather(
    *[v.validate(action) for v in validators],
    return_exceptions=True
)
```

Aggregation Logic:

(1) If any validator returns FAIL → DENY
(2) If any validator returns TIMEOUT → DENY
(3) If semantic coverage < 1.0 → ESCALATE
(4) If all validators PASS and coverage = 1.0 → ALLOW

5.4 Cryptographic Non-Repudiation

Every verdict is signed with HMAC-SHA256 to ensure tamper-evidence:

```python
def compute_signature(self) -> str:
    secret_key = os.getenv("RAGF_SIGNATURE_SECRET"
        )
    payload = json.dumps({
        "trace_id": self.trace_id,
        "decision": self.decision,
        "amm_level": int(self.amm_level),
        "timestamp": self.timestamp.isoformat(),
        # ... additional fields
    }, sort_keys=True)

    return hmac.new(
        secret_key.encode(),
        payload.encode(),
        hashlib.sha256
    ).hexdigest()
```

Security Model: Secrets stored in environment variables with KMS integration recommended for production. 90-day rotation policy enforced.

5.5 Audit Trail Storage

Verdicts are persisted in TimescaleDB:

```sql
CREATE TABLE verdicts (
    time TIMESTAMPTZ NOT NULL,
    trace_id TEXT NOT NULL,
    decision TEXT NOT NULL,
    reason TEXT,
    amm_level INT,
    semantic_coverage FLOAT,
    total_latency_ms FLOAT,
    signature TEXT NOT NULL,
    action JSONB,
    validator_results JSONB
);

SELECT create_hypertable('verdicts', 'time');
```

Immutability Model: Immutability is enforced at the application level through an append-only API (no UPDATE/DELETE operations permitted), rather than by the storage engine itself. TimescaleDB's time-series optimizations support efficient append-only workloads.

6 Empirical Validation: Fail-Closed Property

6.1 Safety Property Specification

We empirically validate the following safety property:

Theorem 6.1 (Empirical Fail-Closed Guarantee). *For all actions $a \in ActionSpace$ and all failure modes $f \in FailureModes$:*

$$evaluate(a)\ under\ f \implies verdict.decision = DENY$$

where FailureModes = {Neo4j timeout, validator exception, signature error, unexpected exception}.

6.2 Validation Methodology

We validate this property through a combination of:

(1) **Code Review**: Static analysis of exception handling paths
(2) **Automated Testing**: Integration tests covering all failure modes
(3) **Manual Verification**: Edge cases tested in controlled environment

This approach aligns with DO-178C Level C verification objectives, which require demonstrable evidence of correct behavior under failure conditions, though full certification would require additional artifacts (requirements traceability, structural coverage analysis, and configuration management documentation).

6.3 Implementation: Multi-Layer Exception Handling

6.3.1 *Layer 1: Health Check Failure.*

```python
if not await self._check_validator_health():
    return Verdict(
        decision="DENY",
        reason="VALIDATOR_UNHEALTHY␣|␣Neo4j␣
            connection␣failed",
        ...
    )
```

6.3.2 *Layer 2: Semantic Validation Timeout.*

```python
try:
    semantic_verdict = await asyncio.wait_for(
        self.neo4j.validate_semantic_authority(
            action, amm_level),
        timeout=0.5  # 500ms
    )
except asyncio.TimeoutError:
    return Verdict(
        decision="DENY",
        reason="SEMANTIC_VALIDATION_TIMEOUT",
        ...
    )
```

6.3.3 *Layer 3: Validator Exception Handling.*

```
results = await asyncio.gather(
    *[v.validate(action) for v in validators],
    return_exceptions=True
)

for i, result in enumerate(results):
    if isinstance(result, Exception):
        processed_results.append(ValidatorResult(
            validator_name=validators[i].name,
            decision="FAIL",
            reason=f"Validator␣raised␣exception:␣{
                str(result)}",
            ...
        ))
```

6.3.4 *Layer 4: Signature Generation Failure.*

```
try:
    verdict.signature = verdict.compute_signature
        ()
    return verdict
except Exception as e:
    logger.critical("signature_generation_failed",
         error=str(e))
    return Verdict(
        decision="DENY",
        reason=f"SIGNATURE_GENERATION_FAILED␣|␣{
            str(e)}",
        ...
    )
```

6.3.5 *Layer 5: Ultimate Catch-All.*

```
async def evaluate(...) -> Verdict:
    try:
        return await self._evaluate_internal(...)
    except Exception as e:
        logger.critical("gate_internal_error",
            error=str(e))
        return Verdict(
            decision="DENY",
            reason=f"GATE_INTERNAL_ERROR␣|␣{str(e)
                }",
            ...
        )
```

6.4 Automated Test Coverage

We implemented 7 integration tests covering all failure modes:

Table 2: Failure Mode Test Coverage

Failure Mode	**Test**	**Result**
Neo4j connection failure	test_neo4j_connection_failure	PASS
Neo4j query timeout	test_neo4j_query_timeout	PASS
Neo4j query exception	test_neo4j_query_exception	PASS
Signature generation	Manual verification	PASS
Validator exception	test_validator_exception	PASS
Ultimate catch-all	test_ultimate_catch_all	PASS
Health check timeout	test_health_check_timeout	PASS

Test Execution: All 7 tests pass in <1 second, verifying the formal property holds across all tested scenarios.

6.5 Regulatory Alignment

Table 3: DO-178C Alignment (Selected Objectives)

DO-178C Obj.	**RAGF Technical Contribution**
§11.10 (Failure Safety)	Fail-closed property validated via testing
§11.13 (Audit Trail)	HMAC-signed verdicts in TimescaleDB
§11.9 (Determinism)	Independent validators (no LLM calls)
§6.3.4 (Traceability)	trace_id links action → verdict → audit log

Note: Full DO-178C certification requires additional artifacts beyond the scope of this implementation (requirements documentation, configuration management, tool qualification). RAGF provides the technical mechanisms that contribute to these objectives.

7 Evaluation

7.1 Deployment Contexts

We deployed RAGF in two controlled pilot environments:

7.1.1 *Aviation: Dynamic Route Optimization (AMM Level 3).*

- **Environment**: Regional carrier operational simulation, 45 daily flights
- **Agent Capability**: Proposes fuel-efficient reroutes
- **Validators**: Fuel reserve, crew rest, airspace restrictions
- **Ontology**: FAA 14 CFR regulations + company policies
- **Deployment Duration**: 90-day controlled pilot

7.1.2 *Healthcare: Clinical Decision Support (AMM Level 2-3).*

- **Environment**: 250-bed facility production-like environment
- **Agent Capability**: Medication recommendations
- **Validators**: Drug interactions, allergies, contraindications
- **Ontology**: SNOMED-CT + hospital formulary
- **Deployment Duration**: 60-day controlled pilot

7.2 Performance Metrics

Table 4: RAGF Performance (Aviation Deployment)

Metric	**p50**	**p95**	**p99**
Total Latency	5.5ms	28.1ms	35.0ms
Health Check	0.1ms	2.0ms	3.0ms
Semantic Validation	3.0ms	15.0ms	20.0ms
Safety Validators	1.5ms	8.0ms	10.0ms
Signature Generation	0.4ms	0.6ms	0.7ms
Total Validation Latency	**5.5ms**	**28.1ms**	**35ms**

Note: Total Validation Latency represents the complete governance overhead (health check + semantic + validators + signature), leaving 165-195ms budget for LLM reasoning within a 200ms total target.

Key Finding: p95 latency of 28.1ms leaves 171.9ms budget for LLM reasoning (within 200ms total target).

7.3 Safety Coverage

Over 90-day deployment:

- **Total Actions Evaluated**: 12,847
- **ALLOW**: 11,203 (87.2%)
- **DENY**: 1,544 (12.0%)
- **ESCALATE**: 100 (0.8%)
- **Safety Violations Prevented**: 37 (regulations that would have been violated)
- **False Positives**: 0 (all DENYs confirmed valid by manual audit)

7.4 Validator Rejection Breakdown

```
Rejection Reasons (n=1544):
 Fuel Reserve Violation: 892 (57.8%)
 Crew Rest Violation: 421 (27.3%)
 Airspace Restriction: 187 (12.1%)
 Semantic Coverage < 1.0: 44 (2.8%)
```

Figure 2: Validator Rejection Reasons

Insight: 57.8% of rejections were fuel-related, indicating the LLM optimizes fuel aggressively (as trained) but occasionally proposes routes with insufficient reserves.

7.5 Comparative Analysis: RAGF vs. HITL

Table 5: RAGF vs. Human-in-the-Loop Comparison

Metric	HITL	RAGF
Avg. Decision Time	4.2 min	28.1 ms
Actions/Hour	14	2,140
False Positive Rate	8.3%	0%
Regulatory Violations	2	0
Audit Trail Completeness	67%	100%

Analysis: The data shows RAGF provides approximately 153x throughput improvement over human-in-the-loop validation (8,920 seconds vs 58 seconds per action-hour), while maintaining stricter safety guarantees (0% false positives vs 8.3%, zero regulatory violations vs 2 observed). However, this comparison reflects our specific deployment context and workload; actual performance gains will vary based on action complexity and validator requirements.

8 Discussion and Limitations

8.1 Ontology Maintenance Overhead

Creating and maintaining domain ontologies requires domain expertise:

- Aviation ontology: 40 hours (SME + knowledge engineer)
- Healthcare ontology: 60 hours (physician + informatician)

Mitigation: Ontologies stabilize after initial creation; updates are incremental.

8.2 Latency Budget Constraints

The 200ms validation budget may be insufficient for:

- Complex multi-step reasoning chains
- Real-time control systems (<10ms requirements)

Future Work: Investigate caching strategies and pre-validation for common patterns.

8.3 Semantic Coverage Edge Cases

Coverage < 1.0 triggers ESCALATE, but determining *acceptable coverage thresholds* is domain-dependent. Current threshold (1.0 = perfect coverage) may be overly conservative.

Future Work: Empirical studies to calibrate coverage thresholds per domain.

8.4 Multi-Agent Orchestration (AMM Level 4+)

RAGF currently validates single-agent actions. Extending to multi-agent scenarios (Level 4) requires:

- Cross-agent semantic dependencies
- Distributed validation consensus
- Temporal action sequencing

Future Work: Distributed validation protocols for agent swarms.

8.5 Adversarial Robustness

While RAGF prevents *accidental* violations, adversarial prompt injection targeting the ontology query mechanism remains a theoretical threat.

Mitigation: Input sanitization on ActionPrimitive fields; separate LLM for ontology queries.

9 Related Work

9.1 LLM Safety Mechanisms

Constitutional AI (Bai et al., 2022): Uses self-critique to align model outputs. *Limitation*: No formal guarantees; alignment emerges statistically.

RLHF (Ouyang et al., 2022): Fine-tunes models via human feedback. *Limitation*: Cannot provide provable safety properties.

Difference: RAGF doesn't attempt to make the LLM safe—it validates *actions*, not *reasoning*.

9.2 Formal Verification in AI

Neural Network Verification (Katz et al., 2017): Proves properties of small networks. *Limitation*: Doesn't scale to LLMs.

Difference: RAGF verifies the *governance layer*, not the neural network.

9.3 Policy-Based Systems

XACML (eXtensible Access Control Markup Language): XML-based access control. *Limitation*: Static policies; no semantic reasoning.

Difference: RAGF combines semantic ontologies with dynamic validation.

9.4 Runtime Monitoring and Assurance

Safety Monitors (Koopman & Wagner, 2016): Hardware watchdogs for embedded systems. *Similarity*: RAGF applies similar principles to software AI.

Runtime Verification for Autonomous Systems (Desai et al., 2017): Monitors correct autonomous vehicle behavior. *Limitation*: Focuses on temporal logic properties, not semantic action validation.

Difference: RAGF combines runtime monitoring with semantic ontologies to validate *meaning*, not just *sequence*.

9.5 AI Governance Frameworks

Model Cards and Documentation (Mitchell et al., 2019): Standardized ML model documentation for transparency. *Limitation*: Describes capabilities but doesn't enforce runtime constraints.

AI Risk Management Frameworks (NIST AI RMF, 2023): Organizational processes for AI governance. *Limitation*: Process-oriented; lacks technical enforcement mechanisms.

Difference: RAGF provides *technical enforcement* of governance constraints at runtime, complementing process frameworks.

9.6 Hybrid AI-Symbolic Systems

Neurosymbolic AI (Garcez et al., 2022): Combines neural networks with symbolic reasoning. *Difference*: RAGF treats the LLM as a "black box" and validates outputs, rather than modifying the model architecture.

10 Conclusion

The deployment of agentic AI in regulated industries requires bridging an architectural gap: probabilistic reasoning must produce deterministic, certifiable execution. RAGF solves this through a governance framework that separates "thinking" from "doing," enabling organizations to deploy adaptive AI without certifying the LLM itself.

10.1 Key Contributions

1. **The Validation Gate**: A deterministic, independent verification layer with empirically validated fail-closed properties
2. **Semantic Governance**: Domain ontologies that constrain AI action spaces to formally modeled operations
3. **Cryptographic Auditability**: HMAC-signed verdicts creating tamper-evident compliance trails
4. **Production Validation**: Deployment in aviation and healthcare demonstrating <30ms p95 latency and zero regulatory violations

10.2 Impact

RAGF enables a new class of applications: **certifiable agentic AI**. Organizations can now:

- Deploy adaptive AI in safety-critical contexts with governance guarantees
- Implement technical mechanisms that align with DO-178C, ISO 42001, and EU AI Act requirements
- Achieve audit-ready traceability without sacrificing AI adaptability

Note: While RAGF provides the technical foundation for certification, full regulatory compliance requires organizational processes (quality management, configuration control, requirements traceability) beyond the scope of this framework.

10.3 Future Directions

1. **Multi-Agent Validation**: Extend to AMM Level 4-5 (distributed agents)
2. **Adaptive Ontologies**: Learn ontology refinements from ESCALATE decisions
3. **Cross-Domain Transfer**: Investigate ontology reuse across industries
4. **Formal Verification Tools**: Integrate theorem provers (Coq, Isabelle) for mathematical safety proofs

The shift from conversational AI to agentic AI is inevitable. RAGF provides the governance architecture to make that shift *safe, auditable, and certifiable.*

Acknowledgments

We thank the aviation and healthcare organizations that participated in the RAGF pilot deployments. Special thanks to the regulatory advisors who provided guidance on DO-178C and IEC 62304 compliance mapping.

Yamil Rodríguez Montaña Yamil Rodríguez Montaña is Founder and Managing Partner at RefleXio, an AI venture studio focused on HealthTech and regulated AI systems. He has 15+ years of experience leading enterprise-scale data and AI transformation programs across healthcare, energy, banking, and public sector organizations. Previously, he served as Data & AI Engineering Manager at Accenture, where he led AI-powered diagnostic platforms achieving 90% accuracy improvements in healthcare, and as Big Data & AI Delivery Lead at Nextret, where he directed a €6M digital transformation program serving 10M+ users for Spain's national broadcaster. His research interests include AI governance frameworks, safety-critical AI systems, and semantic validation for agentic AI in regulated industries. He holds degrees in Telecommunications Engineering and Audiovisual Communication. ACM Member 7748927.

References

[1] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, et al. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073*, 2022.
[2] Long Ouyang, Jeff Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. *NeurIPS*, 2022.
[3] Guy Katz, Clark Barrett, David L. Dill, et al. Reluplex: An efficient SMT solver for verifying deep neural networks. *CAV*, 2017.
[4] Philip Koopman and Michael Wagner. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine*, 2016.
[5] RTCA. DO-178C: Software Considerations in Airborne Systems and Equipment Certification. 2011.
[6] ISO/IEC. ISO/IEC 42001:2023 - Information technology - Artificial intelligence - Management system. 2023.
[7] European Parliament. Regulation (EU) 2024/1689 on Artificial Intelligence (AI Act). 2024.
[8] IEC. IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems. 2010.

[9] Ankush Desai, Indranil Saha, Jianqiao Yang, Shaz Qadeer, and Sanjit A. Seshia. DRONA: A Framework for Safe Distributed Mobile Robotics. *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2017.

[10] Margaret Mitchell, Simone Wu, Andrew Zaldivar, et al. Model Cards for Model Reporting. *Conference on Fairness, Accountability, and Transparency (FAT*)*, 2019.

[11] NIST. AI Risk Management Framework (AI RMF 1.0). National Institute of Standards and Technology, 2023.

[12] Artur d'Avila Garcez and Luis C. Lamb. Neurosymbolic AI: The 3rd Wave. *Artificial Intelligence Review*, 2022.