A Comparative Evaluation of AI Agent Orchestration Frameworks for Regulated Environments

Renzo Zukeram, Vinícius Mergulhão, Yuri de Medeiros, Marcantoni Melo, Ramiro Santos Júnior, Stefano Loss, Nélio Cacho, Frederico Lopes

[1]Federal University of Rio Grande do Norte, Natal, Brazil

renzo.zukeram.705@ufrn.edu.br, vbarbosa.mg@gmail.com
yurimedeiros141@gmail.com, marcantoni.melo.017@ufrn.edu.br
ramirovsjunior@gmail.com, momoloss10@gmail.com
neliocacho@gmail.com, fred@imd.ufrn.br

Abstract. *Organizations operating under compliance mandates increasingly rely on AI agents to automate workflows involving unstructured documents and dynamic decision-making. In such settings, agentic systems must reconcile autonomy with strict requirements for auditability, controlled variability, and integration with legacy infrastructures. We propose an Exemplar as a reusable evaluation artifact for analyzing agentic orchestration frameworks under Compliance Management and Documentary Uncertainty. The Exemplar captures the interaction between probabilistic extraction and deterministic constraint enforcement in workflows characterized by unstructured inputs, rigid schemas, and asynchronous processes. We empirically evaluate four frameworks (CrewAI, Embabel, LangChain, and n8n) across three dimensions: type safety, reasoning auditability, and legacy interoperability. The evaluation is grounded in a real-world instantiation within a state penitentiary agency, involving automated processing of employment contracts with incomplete data and asynchronous stakeholder interactions. Results show clear trade-offs between flexibility and control, highlighting the importance of hybrid approaches that combine agentic reasoning with deterministic validation. The exemplar generalizes to regulated domains such as public procurement, finance, and healthcare.*

1. Introduction

The transition from traditional large language models (LLMs) to AI agents is marked by a shift from passive text generation to iterative perception–action cycles [Wang et al. 2024, Russell et al. 1995]. In this paradigm, models act as mediators that transform observations into tool invocations and verifiable system states, enabling interaction with external environments [Krishnan 2025]. Recent advances (2023–2025) highlight a move toward cognitive autonomy, where agents integrate context perception, reasoning over unstructured data, and goal-directed decision-making [Liu et al. 2023, Park et al. 2023].

However, this autonomy introduces new risks. Unlike standalone LLMs, agentic systems can trigger irreversible actions in real-world systems, making stochastic errors unacceptable in regulated environments [Luo et al. 2025, Li and Ning 2023]. This tension is particularly evident in administrative workflows, where unstructured inputs must be reconciled with rigid legal and operational constraints.

To address this gap, we propose an *Exemplar* as a reusable[1] evaluation artifact for agentic orchestration frameworks under *Compliance Management and Documentary Uncertainty*. It captures the interaction between probabilistic extraction and deterministic validation in workflows from unstructured documents, strict schemas, and asynchronous processes. By design, it enables systematic evaluation of how different orchestration approaches balance flexibility, auditability, and integration with legacy systems.

Building on this foundation, we conduct a comparative analysis of four orchestration frameworks (CrewAI, Embabel, LangChain, and n8n) drawing on both architectural insights (e.g., memory, planning, and tool use) and empirical evidence from multi-agent systems research [Wang et al. 2024, Luo et al. 2025, Wu et al. 2024]. The evaluation focuses on three critical dimensions for regulated environments: type safety, reasoning auditability, and interoperability with legacy infrastructures. Accordingly, this study is guided by the following Research Questions (RQs):

1. RQ1 (Determinism and Integrity): *How do distinct orchestration architectures, ranging from purely stochastic to neuro-symbolic models, impact data integrity and type safety when extracting structured information from unstructured legal documents?*
2. RQ2 (Auditability): *To what extent do the evaluated frameworks provide native mechanisms for reasoning transparency and state persistence necessary to meet public and private sectors accountability standards?*
3. RQ3 (Integration with Legacy Systems): *What trade-offs exist between the rapid prototyping capabilities of low-code or Python-centric frameworks and the interoperability requirements of legacy enterprise Java ecosystems?*

This paper addresses these RQs through three key contributions. First, we propose an exemplar capturing the interplay among probabilistic extraction, deterministic validation, and asynchronous orchestration. Second, we systematically evaluate four agentic frameworks across type safety, auditability, and legacy interoperability. Third, we demonstrate generalizability via a real-world instantiation in a state penitentiary agency, highlighting practical mission-critical trade-offs.

2. Background

2.1. The Engineering Reality of Multi-Agent Orchestration

According to Sapkota et al. 2025, AI Agents consist of autonomous and modular software entities, frequently driven by Large Language Models (LLMs), designed to execute specific tasks through sequential reasoning and external tool use. In contrast, Multi-Agent Systems (MAS) represent an architectural approach based on the collaboration of multiple specialized agents. Within these ecosystems, dynamic task decomposition, persistent memory sharing, and continuous communication occur, enabling an orchestrated autonomy capable of solving highly complex, interdependent problems that would be unfeasible for a single isolated agent [Sapkota et al. 2025].

The primary challenge in multi-agent systems resides in the orchestration layer, which governs the transition from experimental scripts to production-ready deployments.

[1]Used codes available at: https://github.com/renzozuk/AI-Agents-Implementation

Frameworks address this complexity by externalizing collaboration logic: *CrewAI* structures interactions through hierarchical role delegation, whereas *n8n* embeds artificial intelligence capabilities within predefined deterministic workflow nodes [Derouiche et al. 2025, Pelluru 2025, Duan and Wang 2024].

2.2. Taxonomy of Modern Agentic Frameworks

Many frameworks are available today, but the four selected (Table 1) were chosen for their relevance and impact across the AI development stack. At one end, workflow orchestrators (n8n) and LLM libraries (LangChain) prioritize integration and quick deployment within existing stacks [Bannett 2024, Jeong et al. 2024]. At the other end, state-machine architectures and neuro-symbolic approaches (Embabel) treat execution as a deterministic planning problem. Embabel, for instance, is particularly relevant for high-stakes administrative environments: it combines LLM flexibility with Goal-Oriented Action Planning (GOAP), allowing it to veto risky plans before execution [Luong et al. 2024].

Table 1. Frameworks Considered in this Study and Their Technical Specificities.

Framework	Primary scope	Orchestration model	State / memory handling	Auditability & safety primitives	Typical use context
CrewAI	Role-based MAS	Hierarchical crews; task delegation	Structured memory (short/long/entity)	Multi-role critique; guardrails	Collaborative analysis [Duan and Wang 2024]
Embabel	Neuro-symbolic	GOAP planning	Context-as-typed-state; dynamic KB	A priori plan inspection; vetos	Compliance-heavy tasks [Luong et al. 2024]
LangChain	LLM App Lib	Chains and tool-calling policies	Composable memory/retrievers	Prompt/tool validation patterns	Rapid prototyping [Jeong et al. 2024]
n8n	Workflow automation	Visual nodes; JS/Python extensible	Workflow persistence; API-driven state	Execution logs; HITL approval steps	Enterprise automation [Bannett 2024]

The technical landscape outlines the current *how* of agent orchestration, but these tools do not exist in a vacuum. Their design divergence (from structured, deterministic flows to fluid, role-playing crews) reflects a broader debate in the literature on balancing agentic freedom with operational predictability. To position our study within this evolving field, we must move beyond these engineering primitives and examine how the academic community has historically tackled the challenges of agent evaluation, safety, and multi-component coordination.

2.3. Regulated Environments

Regulated environments are characterized by persistent operational challenges, including process fragmentation, data format incompatibility, and the rigidity of administrative workflows [Mequanenit et al. 2025]. The implementation of multi-agent systems (MAS) in these scenarios aims to enhance efficiency and the capacity to solve complex problems. However, the application of autonomous and collaborative agents requires effective governance infrastructures and meticulous management of human-AI interactions, given the ethical and operational complexities involved [Zhang et al. 2026, Mequanenit et al. 2025].

The choice of regulated environments as the setting for this analysis is based on the need to balance agent autonomy with strict compliance. These domains serve as a 'stress test' for frameworks, as they require AI reasoning to be flexible for problem-solving while being fully auditable and secure. Unlike general-purpose applications, the complexity of these sectors allows for evaluating how tools manage unstructured data flows, revealing technical limitations that emerge only under such constraints.

3. Related Works

Although research on Agentic AI has rapidly expanded, public and private sectors adoption remains constrained less by raw model capability and more by architectural guarantees around control, traceability, and integration [Derouiche et al. 2025]. Thus, in compliance-critical workflows, the central challenges are (i) preserving data integrity under uncertainty, (ii) producing auditable traces of multi-step reasoning and tool use, and (iii) integrating safely with legacy systems that enforce strict operational boundaries.

Compliance in regulated environments requires strict integrity and auditability guarantees. While Agentic AI is recognized for its iterative reasoning and planning capabilities [Salah et al. 2025], its practical validation faces significant methodological obstacles. Current evaluation practices suffer from a systemic imbalance: technical metrics dominate 83% of analyses, while safety (53%) and human factors (30%) remain peripheral, a gap that undermines industrial productivity claims in mandatory compliance sectors [Meimandi et al. 2025]. This justifies a comparative analysis of orchestration frameworks (CrewAI, Embabel, LangChain, and n8n) focusing on their ability to maintain the security and auditability required for sensitive workflows.

Beyond single-agent designs, multi-agent systems are often proposed to decompose complex workflows into specialized roles. Derouiche et al. 2025 survey architectures and protocols underpinning *Agentic AI*, as adoption bottlenecks increasingly involve orchestration robustness and operational security rather than model capacity alone. Along similar lines, AutoGen operationalizes multi-agent workflows as structured conversations among specialized agents [Wu et al. 2024]. However, conversation-based coordination alone does not guarantee termination, consistency, or state correctness in multi-step tasks.

In high-stakes domains, recent work explores hybrid controls to reduce stochastic failures when mapping unstructured inputs to structured outputs. Luong et al. 2024 proposes domain-aware neuro-symbolic agents aimed at improving consistency and accuracy by injecting explicit structure and constraints into agent behavior. This applies directly to regulated administrative settings, where an agent must avoid producing legally invalid states even when source documents are incomplete or ambiguous.

Collectively, these studies establish techniques for controlled reasoning, multi-agent collaboration, and hybrid constraints for structured output consistency. However, they often stop short of comparing orchestration frameworks under a single, compliance-oriented evaluation scheme that makes data integrity/type safety, auditability/state persistence, and legacy interoperability explicit and measurable. This gap motivates the comparative criteria and the public-sector case study adopted in this paper.

4. Methodology

To assess the Agentic Frameworks under analysis, we designed and implemented a well-structured exemplar. According to [Sim et al. 2003], an "exemplar" is a reference application or a collection of tests used to compare the performance of alternative software engineering tools or techniques. Unlike a "benchmark," an exemplar is more flexible, as it does not enforce specific metrics or predefined performance criteria for comparison. This flexibility makes exemplars particularly suitable for exploring the characteristics and capabilities of emerging tools and techniques. Furthermore, Sim emphasizes that the development of both exemplars and benchmarks plays a key role in driving technology

transfer and advancing research in software engineering [Sim et al. 2003]. Figure 1 illustrates the processing pipeline defined in the proposed exemplar, which operates through a structured schema validation process and comprises four sequential orchestration phases:

- **Ingestion and Structured Extraction:** The process begins with the vector ingestion of unstructured documents (PDFs), extracting and projecting their content onto a target JSON schema. In this stage, we prioritize type safety to ensure that the transition from raw text to structured data maintains total technical fidelity.
- **Deterministic Gap Validation:** The system performs an automated integrity audit on the generated JSON object to identify omissions or inconsistencies, such as malformed monetary values or missing fiscal identifiers. Unlike purely probabilistic approaches, this phase imposes rigid business logic (hard constraints) to ensure compliance.
- **Proactive Communication and Data Recovery:** Upon detecting informational gaps, the agent triggers, via Tool Use, an external interaction loop. It generates and sends formal inquiries to stakeholders, automating the search for complementary data without requiring synchronous human intervention.
- **State Persistence and Asynchronous Reactivation:** The flow manages the application state through a stateless execution model. Instead of holding suspended processes in memory, the agent persists the partial context in a database and terminates execution. Resumption occurs via API triggers (webhooks), which reinject the new data to consolidate the final document.

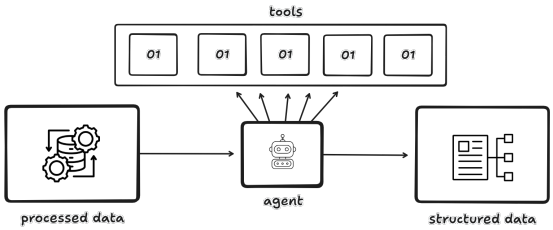

Figure 1. Autonomous agent processing, from processed vector data to a validated structured format through dynamic tool orchestration.

This exemplar establishes a controlled experimental setting to evaluate orchestration platforms' ability to handle state persistence and maintain information integrity in regulated environments. By concretely instantiating the end-to-end workflow, it enables systematic observation and comparison of how different frameworks manage asynchronous execution, enforce constraints, and recover incomplete data.

5. The Exemplar Description

This section introduces the proposed exemplar, designed to capture a class of public administration problems characterized as *Compliance Management and Documentary Uncertainty*. These problems involve the interplay of unstructured data ingestion, validation against rigid legal schemas, and asynchronous process orchestration: challenges prevalent in domains such as tax regulation, public health, and public procurement.

The exemplar operationalizes these challenges into a concrete and reusable evaluation artifact for agentic orchestration frameworks. Rather than focusing on isolated metrics, it captures the interaction between probabilistic extraction and deterministic constraint enforcement under real-world conditions. This enables a systematic assessment

of how such frameworks reconcile the variability of unstructured inputs with the strict requirements of authoritative systems of record. While grounded in public administration, the exemplar is directly applicable to regulated domains such as financial services, healthcare, and infrastructure management [Wirtz et al. 2019].

To instantiate this contribution, the exemplar models the administrative workflow of a state penitentiary agency managing inmate custody and private-sector labor agreements. Within this setting, the exemplar implements four sequential orchestration phases: ingestion and structured extraction, deterministic gap validation, proactive communication and data recovery, and state persistence with asynchronous reactivation. This workflow exposes critical technical challenges, including processing heterogeneous, unstructured PDF contracts and handling missing identifiers, like Corporate Taxpayer IDs (CNPJ), which require asynchronous stakeholder interaction. Given that these processes involve sensitive financial and legal operations under institutional constraints, the exemplar imposes strict integrity and auditability requirements, directly exposing the limitations of purely stochastic approaches.

Figure 2 illustrates the workflow implemented in the proposed exemplar, whose purpose is to identify and allocate job opportunities for inmates participating in resocialization programs. In this context, the system leverages a public API that aggregates job openings from companies contracted by the government (e.g., for infrastructure projects such as school construction). By legal mandate, these companies must reserve a percentage of positions for inmates. The exemplar operationalizes this requirement by identifying eligible opportunities and supporting the negotiation process to match candidates with the most suitable profiles for each available position.

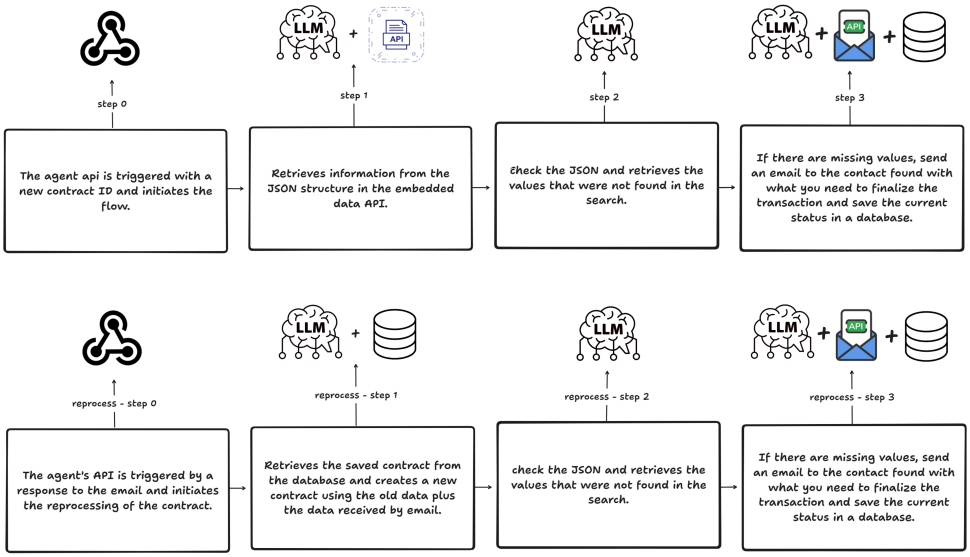

Figure 2. Agent orchestration workflow for extracting, validating, and retrieving contractual data with support for asynchronous human-in-the-loop interactions.

The workflow proceeds as follows. In Step 0, the process is triggered via an API call when a new contract identifier is received, initiating the agentic workflow. In Step 1, the system performs ingestion and structured extraction by retrieving relevant information from the embedded data API and mapping it into a predefined JSON schema. In Step 2,

a deterministic validation phase checks the generated JSON object, identifying missing or inconsistent values that were not captured during extraction. In Step 3, if gaps are detected, the agent initiates an external communication loop by sending a request (e.g., email) to the appropriate stakeholder and persists the current state in a database.

The lower part of the figure represents the asynchronous reprocessing cycle. In Reprocess Step 0, the system is reactivated upon receiving a response (e.g., via webhook or API trigger). In Reprocess Step 1, the agent retrieves the stored contract state and merges it with the new information. In Reprocess Step 2, the validation process is executed again to ensure completeness and consistency. Finally, in Reprocess Step 3, if necessary, additional communication cycles are triggered, and the updated state is persisted until all required information is obtained and the allocation process can be finalized.

6. Experimental Settings

The experimental corpus consists of PDF labor contracts between the studied public agency and its private-sector partners. These documents exhibit significant structural variability, particularly in remuneration formats, fiscal identifiers (e.g., CPF and CNPJ), and job descriptions. A typed validation schema was defined in collaboration with the agency's technical staff to ensure alignment with institutional requirements. While a systematic quantitative characterization of the corpus is not provided, representing a known limitation, this aspect is further discussed in Section 8.

The workflow follows the proposed exemplar pipeline. The agent retrieves document content from a vector store, maps it to typed data transfer objects (DTOs), and validates mandatory attributes through prompt-guided consistency checks. When inconsistencies or missing fields are detected, the system initiates an asynchronous human-in-the-loop (HITL) process: the agent issues a stakeholder request, persists its state, and suspends execution. Processing resumes upon webhook-triggered reactivation, where newly acquired data is merged and revalidated. This cycle repeats until full schema compliance is achieved, after which the validated record is submitted to the contracts API.

All evaluated frameworks execute the same pipeline over identical datasets, schemas, and LLM configurations, ensuring that observed differences stem from orchestration strategies rather than experimental conditions. The technical characteristics of Embabel (such as GOAP-based planning, backward chaining, and native JVM integration) are documented in [Luong et al. 2024] and were directly tested during the experiments. The classifications reported in Section 7 are therefore based on empirical observations rather than solely on prior literature.

7. Results and Discussion: Comparative Analysis of Agentic Frameworks

7.1. Evaluation Framework and Metrics Matrix

Aligned with RQ1–RQ3, this study evaluates framework performance across three dimensions: schema-compliant output rates for type safety (RQ1), execution trace granularity, and checkpoint recovery for operational transparency (RQ2), and workload and latency in legacy Java-based ecosystems for enterprise integration (RQ3). We derived the classifications presented in Table 2 from a static analysis of the integration code, an inspection of execution artifacts, including logs, traces, and persisted state objects, and a comprehensive review of the official documentation for each platform. The resulting classifications

should be interpreted as indicative assessments grounded in the authors' direct operational experience, rather than as statistically conclusive findings. The absence of independent inter-rater validation, acknowledged as a threat to internal validity, bounds the generalizability of these ordinal scores to the experimental context described.

All classifications in Table 2 derive from the empirical assessment described in Section 6. The frameworks were evaluated across eight specific criteria using a consistent four-level ordinal scale: *Very High*, *High*, *Medium*, and *Low*. For instance, regarding *Type Safety*, *Very High* denotes runtime rejection of schema-violating objects prior to any persistence, while *Low* denotes silent error propagation. The resulting matrix categorizes the frameworks by revealing critical trade-offs between *Developer Experience* indicators and *Operational Guarantees*, serving as the empirical baseline for the detailed architectural fit discussion provided in the subsequent section.

Table 2. Architectural Capabilities and Developer Experience Comparison

Evaluation Criteria	n8n	CrewAI	LangChain	Embabel
Implementation Simplicity	Very High	Medium	Medium	Low
Boilerplate Code	Very Low	Moderate	Moderate	High
Prompt Dependence	High	High	High	Medium
Type Safety (Schema Adherence)	Very Low	High	Medium	Very High
Reasoning Auditability	Low (Visual)	Medium (Logs)	Medium (Tracing)	High (GOAP)
Fail-Fast Behavior	Low	Medium	Medium	High
Legacy Integration	API-only	Python Bridge	Python Bridge	Native (JVM)
Long-running State Support	Native (Visual)	Manual	Manual	DB-Backed

Starting with *Developer Experience*, *Implementation Simplicity* measures the learning curve, structural clarity, and coding effort required to implement basic features. This criterion is most accessible in n8n, rated *Very High*, while CrewAI and LangChain are rated *Medium*, and Embabel is rated *Low*. Conversely, *Boilerplate Code* evaluates the amount of repetitive setup, annotations, conventions, and auxiliary code required to use the framework effectively. In this regard, n8n demands the least boilerplate, the Python-based frameworks require a moderate amount, and Embabel requires a higher level of structured code. Regarding cognitive execution, *Prompt Dependence* refers to the degree to which output reliability depends on extensive prompt engineering. This dependency is *High* across n8n, CrewAI, and LangChain, but only *Medium* for Embabel, reflecting its neuro-symbolic design.

In terms of *Operational Guarantees* for reliability-sensitive workflows, *Type Safety* assesses the ability to ensure returned objects strictly adhere to predefined structures. Under this criterion, Embabel achieves a *Very High* classification, followed by CrewAI with *High*, LangChain with *Medium*, and n8n with *Low*. *Reasoning Auditability*, the ease of tracking intermediate decisions and tool calls, also varies significantly. Embabel provides *High* transparency through GOAP, while CrewAI and LangChain offer *Medium* transparency via logs and tracing. In contrast, n8n is limited to *Low* auditability, primarily due to its visual flow. Similarly, *Fail-Fast Behavior*, which measures the capacity to halt execution before errors propagate, is *High* in Embabel, *Medium* in the Python-based frameworks, and *Low* in n8n.

Finally, infrastructural capabilities reveal additional differences among the frame-

works. *Legacy Integration* assesses the ease of integration with existing internal services, databases, and traditional architectures. Embabel benefits from native JVM integration, whereas CrewAI and LangChain typically rely on Python-based bridges, and n8n is mainly constrained to API-based integrations. For *Long-running State Support*, which is essential for maintaining context, tracking progress, and resuming multi-step asynchronous tasks, Embabel adopts a database-backed approach, while n8n provides native visual workflow management. CrewAI and LangChain, in contrast, generally require manual implementation to support long-running stateful processes.

An assessment of token consumption and financial expenditure, summarized in Table 3, reveals a significant operational divergence between n8n and the agentic frameworks. n8n emerged as the most economical solution (US$ 0.0033 in total), primarily because of its static architecture. By relying on pre-configured logic nodes for routing and state management, n8n restricts LLM usage strictly to information extraction tasks, minimizing Input Tokens to only the essential document context.

Table 3. Per-Framework Token Consumption and Cost Metrics (model: gpt-5-nano; costs extracted from OpenAI API response metadata).

Evaluation Criteria	n8n	CrewAI	LangChain	Embabel
First Flow Input Tokens	1,676	38,137	72,173	21,845
First Flow Output Tokens	1,810	27,355	29,802	24,934
First Flow Total Tokens	3,486	65,492	101,975	46,779
First Flow Cost (USD)	0.0009	0.0129	0.0129	0.0111
Second Flow Input Tokens	6,632	21,200	6,592	17,001
Second Flow Output Tokens	5,102	3,717	14,179	22,501
Second Flow Total Tokens	11,734	24,917	20,771	39,502
Second Flow Cost (USD)	0.0024	0.0026	0.0059	0.0099
Total Cost (USD)	0.0033	0.0155	0.0188	0.0210

In contrast, CrewAI, LangChain, and Embabel delegate orchestration to the LLM, resulting in a substantially higher consumption profile. The high Input Token count in these frameworks, peaking at 101,975 in LangChain, is driven by the need to inject "System Prompts," tool schemas (Pydantic/Java Records), and conversation history into the context window at each interaction step. Furthermore, Embabel's neuro-symbolic nature generated the highest total cost (US$ 0.0210), since its GOAP architecture generates detailed execution plans and reasoning traces, significantly inflating Output Token usage compared to n8n's purely reactive extraction.

7.2. Architectural Trade-offs and Operational Fit

Comparative data reveals a tension between development velocity and operational control. Table 2 indicates an inverse correlation between implementation simplicity and transparency. Low-code platforms like n8n enable rapid prototyping but introduce a black-box effect, where visual abstractions obscure decision logic and hinder accountability in regulated environments. Conversely, Embabel requires extensive configuration but offers superior auditability. Its GOAP algorithm explicitly exposes execution plans before action, enabling deterministic verification of agent behavior.

This trade-off extends to Data Integrity and Type Safety. In legal contexts, hallucination risk is unacceptable. Prompt-dependent frameworks like n8n and LangChain

proved susceptible to structural errors, such as inconsistent JSON keys, during LLM improvisation. Conversely, Embabel and CrewAI enforce strict schema adherence via Java Records and Pydantic. This structural enforcement serves as a programmatic safeguard and a Fail-Fast philosophy: if data format is not guaranteed, the process terminates rather than propagating corrupted data. While LangChain supports Pydantic, its architecture prioritizes flexible prompts, whereas Embabel treats type safety as a first-class citizen.

Finally, the *Legacy Integration* criterion is decisive for organizations with established infrastructure. While Python solutions (CrewAI, LangChain) offer API integration, they introduce architectural overhead. Experiments indicate Embabel enables native JVM integration, allowing agents to directly instantiate Java repositories and libraries within the same memory space, as verified in Section 6. This interoperability suggests that, despite initial costs, the neuro-symbolic approach offers reduced maintenance and seamless integration with mature enterprise ecosystems.

8. Limitations and Threats to Validity

Regarding experimental constraints, the evaluation did not systematically quantify document counts, page length distributions, or clause type coverage. This absence limits statistical interpretability and precise replication, rendering the comparative results indicative rather than conclusive. Additionally, the quantitative assessment in Table 3 focuses exclusively on token consumption and financial costs, omitting pipeline execution latency, which can be a disqualifying factor in time-sensitive workflows. Methodologically, the four-level ordinal scale in Table 2 was derived solely from the authors' code analysis and documentation review. While grounded in the criteria in Section 7.1, the classification lacks independent validation or formal inter-rater agreement, introducing potential observer bias as a threat to internal validity.

Threats to external validity stem from model specification and domain boundaries. Although experiments used the gpt-5-nano[2] family, full snapshot identifiers and API access timestamps were not recorded at runtime, limiting exact reproducibility given continuous model updates. Furthermore, the empirical evaluation was restricted to a single domain (a penitentiary institution). Generalization to other regulated environments (such as tax regulation, public procurement, or healthcare contracts) relies on qualitative similarities in processing requirements rather than empirical validation. Consequently, these domain analogies must be interpreted as hypotheses for future investigations rather than verified results.

9. Conclusions and Future Research

This work presented an empirical comparative evaluation of four AI agent orchestration frameworks (CrewAI, Embabel, LangChain, and n8n) grounded in a compliance-driven workflow in a regulated environment. Central to this study is the proposal of an exemplar as a reusable evaluation artifact, designed to capture the interplay between probabilistic information extraction, deterministic validation, and asynchronous orchestration under conditions of *Compliance Management and Documentary Uncertainty*. The exemplar operationalizes these challenges through a four-phase workflow: vector ingestion, deterministic schema validation, asynchronous communication with stakeholders,

[2]https://developers.openai.com/api/docs/models/gpt-5-nano

and webhook-driven state reactivation. It is instantiated using real-world data from the processing of employment contracts in a state penitentiary agency.

The evaluation assessed frameworks across three critical dimensions: type safety, reasoning auditability, and interoperability with legacy systems, using an ordinal rubric derived from static code analysis, execution artifact inspection, and documentation review. Results highlight clear trade-offs. Embabel achieved the highest scores in compliance-critical dimensions due to its GOAP-based planning and native JVM integration, albeit at the cost of increased implementation complexity and higher token consumption. In contrast, n8n offered the lowest operational cost and fastest prototyping capabilities but exhibited limited type safety and low transparency, making it less suitable for regulated environments. CrewAI and LangChain occupied intermediate positions, with LangChain providing greater flexibility at the expense of structural reliability.

Beyond these empirical findings, the proposed exemplar constitutes a key contribution by providing a controlled yet realistic setting for systematically evaluating agentic orchestration frameworks. It enables the analysis of how such systems enforce constraints, manage asynchronous execution, and ensure data completeness in mission-critical contexts, while remaining generalizable to other regulated domains.

Future work includes: (i) replicating the evaluation across domains with distinct document semantics, such as tax regulation and public procurement, to assess transferability; (ii) extending the analysis to heterogeneous LLM architectures and high-throughput scenarios; (iii) measuring the planning latency of GOAP under concurrent workloads to characterize scalability trade-offs; and (iv) investigating formal verification mechanisms for agentic plans as an extension of neuro-symbolic approaches explored in this study.

Acknowledgements

This work was supported by the SmartMetropolis Project[3] and INES.IA[4], funded by the CNPq under grant 408817/2024-0; it also integrates research activities within the INCT ICoNIoT, likewise supported by CNPq (proc. 405940/2022-0 and 303280/2023-9).

References

Bannett, M. (2024). Deep learning powered architectures for intelligent workflow dynamics, adaptive task scheduling, and autonomous orchestration of complex processes in n8n. *MetaVision Journal of Multidisciplinary Studies (MVJMS)*, 1(3):1–19.

Derouiche, H., Brahmi, Z., and Mazeni, H. (2025). Agentic ai frameworks: Architectures, protocols, and design challenges. *arXiv preprint arXiv:2508.10146*.

Duan, Z. and Wang, J. (2024). Exploration of llm multi-agent application implementation based on langgraph+ crewai. *arXiv preprint arXiv:2411.18241*.

Jeong, J., Gil, D., Kim, D., and Jeong, J. (2024). Current research and future directions for off-site construction through langchain with a large language model. *Buildings*.

Krishnan, N. (2025). Ai agents: Evolution, architecture, and real-world applications. *arXiv preprint arXiv:2503.12687*.

[3]https://smlab.imd.ufrn.br/

[4]www.ines.org.br

Li, Z. and Ning, H. (2023). Autonomous gis: the next-generation ai-powered gis. *International Journal of Digital Earth*, 16(2):4668–4686.

Liu, X., Yu, H., Zhang, H., Xu, Y., et al. (2023). Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Luo, J., Zhang, W., Yuan, Y., Zhao, Y., Yang, J., Gu, Y., Wu, B., Chen, B., Qiao, Z., Long, Q., et al. (2025). Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*.

Luong, V., Dinh, S., Raghavan, S., Nguyen, W., Nguyen, Z., Le, Q., Vo, H., Maegaito, K., Nguyen, L., Nguyen, T., Ha, A. H., and Nguyen, C. (2024). Dana: Domain-aware neurosymbolic agents for consistency and accuracy.

Meimandi, K. J., Aránguiz-Dias, G., Kim, G. R., Saadeddin, L., Griffith, A., and Kochenderfer, M. J. (2025). The measurement imbalance in agentic ai evaluation undermines industry productivity claims. *arXiv preprint arXiv:2506.02064*.

Mequanenit, A. M., Nibret, E. A., Herrero-Martín, P., García-González, M. S., and Martinez-Bejar, R. (2025). A multi-agent deep reinforcement learning system for governmental interoperability. *Applied Sciences*, 15(6):3146.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Pelluru, K. (2025). Langchain & langgraph in production: Architectures for multi-agent llm systems. *Journal of Data and Digital Innovation (JDDI)*, 2(3):1–9.

Russell, S., Norvig, P., and Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25(27):79–80.

Salah, M., Alnoor, A., Abdelfattah, F., and Al Halbusi, H. (2025). Agentic artificial intelligence in public administration: Foundations, applications, and governance. *Applications, and Governance (May 10, 2025)*.

Sapkota, R., Roumeliotis, K. I., and Karkee, M. (2025). Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *Information Fusion*, page 103599.

Sim, S. E., Easterbrook, S., and Holt, R. C. (2003). Using benchmarking to advance research: A challenge to software engineering. In *25th International Conference on Software Engineering, 2003. Proceedings.*, pages 74–83. IEEE.

Wang, L., Ma, C., Feng, X., Zhang, Z., et al. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Wirtz, B. W., Weyerer, J. C., and Geyer, C. (2019). Artificial intelligence and the public sector—applications and challenges. *International Journal of Public Administration*.

Wu, Q., Bansal, G., Zhang, J., et al. (2024). Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

Zhang, R., Liu, G., Liu, Y., et al. (2026). Toward edge general intelligence with agentic ai and agentification: Concepts, technologies, and future directions. *IEEE Communications Surveys & Tutorials*, 28:4285–4318.