algorithms

MDPI

Article

Governed Agentic Process Automation: A Floor-Safety Guarantee for Compliance-Critical LLM Routing

Massimo Pacella [1,*], Gabriele Papadia [1] and Vincenzo Giliberti [2]

[1] Department of Engineering for Innovation, University of Salento, 73100 Lecce, Italy; gabriele.papadia@unisalento.it
[2] IN & OUT S.p.A. a Socio Unico Teleperformance S.E., 74121 Taranto, Italy; vincenzo.giliberti@teleperformance.com
* Correspondence: massimo.pacella@unisalento.it

Abstract

Agentic Process Automation (APA) extends Robotic Process Automation by delegating workflow construction and runtime decisions to Large Language Model (LLM) agents. In regulated business processes, this autonomy creates a specific safety risk: an LLM may recognize elevated risk while still failing to trigger the human approval required by policy. We propose Governed APA, a runtime decision architecture that bounds LLM autonomy through schema-validated inputs, a closed action space, a policy-derived floor guardrail, and a deterministic fallback. We formalize the governed decision and establish a floor-safety property ensuring that the executed action cannot under-escalate below the policy-mandated minimum. The architecture is evaluated on a structured employee-onboarding case study with 166 profiles and explicit human-in-the-loop (HITL) ground truth. We compare a deterministic baseline, an ungoverned LLM, and a guarded LLM using two local models, Qwen2.5 and llama3.1. Ungoverned Qwen2.5 identified elevated risk but failed to escalate any of the 43 sensitive cases to mandatory human approval, yielding HITL recall equal to 0. Ungoverned llama3.1 showed the opposite failure mode, achieving full recall but producing unnecessary human-approval escalations. With the floor guardrail active, Qwen2.5 HITL recall increased to 1.0, llama3.1 recall remained 1.0, and inter-run stability of the compliance decision increased from 0.73 to 1.0 for Qwen2.5. On this dataset and configuration, no false-positive HITL escalation was introduced for Qwen2.5. These results show that compliance-critical APA requires architectural governance rather than prompt-level reliance on LLM discretion. The policy-derived floor guardrail enforces the floor-safety property and prevents silent under-escalation. The guarantee is local: it constrains the compliance routing decision under validated inputs and a trusted policy, and does not cover input corruption, policy errors, extraction failures, prompt injection, downstream tool misuse, or end-to-end workflow correctness. The evaluation is a single-process, two-model proof of concept, and generalization to other regulated workflows is a hypothesis for future validation.

Keywords: agentic process automation; large language model agents; floor-safety; guardrails; human-in-the-loop; employee onboarding; automation bias

Academic Editor: Yousef Farhaoui

Received: 22 June 2026
Revised: 16 July 2026
Accepted: 24 July 2026
Published: 27 July 2026

1. Introduction

Automation aims to reduce human effort in repetitive work. In digital organizations, this aim has been largely implemented through Robotic Process Automation (RPA), which

uses software bots to execute rule-based tasks over existing systems [1–3]. Traditional RPA is effective when processes are stable, structured, and deterministic. Its limits become apparent when workflows must be derived from high-level intent, inputs vary across cases, or execution requires context-sensitive judgment.

Large Language Models (LLMs) have enabled a new approach in this context. Research on LLM-based autonomous agents shows that agents can combine reasoning, planning, memory, tool use, and action [4–6]. In process automation, this shift has motivated Agentic Process Automation (APA), where LLM-based agents construct workflows and make runtime decisions [7]. Related systems embed LLMs into RPA for perception and GUI navigation [8], while proactive agents extend autonomy from command-following behavior to context-aware assistance [9].

This gain in autonomy introduces a specific class of risk. In regulated business processes, some runtime decisions are not ordinary classifications. They determine whether a case must be escalated, reviewed, approved, or logged. If an LLM selects such a route incorrectly, the error may violate policy without triggering an immediate alarm. It may also reinforce automation bias, since users can over-trust automated outputs even when they are wrong [10,11]. This risk differs from ordinary task failure. A visible task failure can be detected during execution, whereas a missed escalation may remain hidden until audit, incident review, or downstream control.

The key issue is, therefore, not only agent capability, but also agent governability. Capability concerns what an agent can construct, perceive, plan, or execute. Governability concerns whether an agent's permitted decisions remain within externally defined organizational constraints. Agentic Business Process Management (BPM) and enterprise-agent research identify the need for guardrails, human oversight, fallback, and accountable integration [12–14]. RPA governance studies stress access control, role clarity, logging, auditability, and change control [15]. LLM safety surveys provide structured safety taxonomies covering attacks, defenses, and guardrails [16]. These streams motivate runtime mechanisms that constrain LLM-selected process actions when policy defines a minimum escalation level.

This paper addresses that need through Governed APA, a constrained-autonomy architecture for compliance-critical routing. The core principle is that the LLM may propose a decision, but policy defines the minimum admissible escalation. When the policy floor is binding, the final governed action cannot fall below it. When the floor is not binding, the LLM retains discretion within the admissible action space.

We study employee onboarding because it is both automatable and compliance-sensitive. HR literature identifies onboarding and related HR operations as natural targets for agentic automation, while also stressing bias, privacy, transparency, and governance risks [17]. In our case, the process involves structured profiles, role-based access provisioning, data-protection requirements, work-mode decisions, and mandatory human approval for sensitive accounts.

The results show that an ungoverned LLM can be silently unsafe on compliance-critical routing. In the Qwen2.5 configuration, the model identifies elevated risk in its generated assessment but fails to trigger mandatory human approval for any sensitive case. A policy-anchored floor guardrail removes this under-escalation failure without replacing the LLM with a deterministic classifier for all cases. The resulting design preserves agentic discretion while enforcing a verifiable compliance floor.

This paper makes the following contributions, summarized conceptually in Figure 1:

- We propose Governed APA, a constrained-autonomy architecture for compliance-critical routing. Each LLM decision is wrapped in four stages: (i) schema-validated structured input, (ii) an LLM decision restricted to a closed action space with validated

output, (iii) a policy-derived guardrail, and (iv) a deterministic fallback. The guardrail is anchored to validated input fields and policy rules represented outside the prompt.
- We provide a compact formal model of the governed decision and the floor guardrail. The model makes the safety property precise. When the policy floor is binding, the governed action cannot fall below the required escalation level, regardless of the LLM proposal.
- We instantiate the architecture on an industrial employee-onboarding case study restricted to structured data. We also provide an evaluation harness that scores each run against explicit human-in-the-loop (HITL) ground truth and measures governance behavior, including guardrail activation, fallback rate, inter-run stability, and baseline versus LLM agreement.
- We provide empirical evidence on 166 onboarding profiles and two local LLMs. Ungoverned Qwen2.5 misses all 43 mandatory HITL cases, while llama3.1 tends to fail in the opposite direction through over-escalation. The floor guardrail raises Qwen2.5 HITL recall from 0 to 1.0 and preserves llama3.1 recall at 1.0, thereby preventing missed mandatory escalation for both models. For Qwen2.5, it does so without introducing spurious HITL escalation on this dataset and configuration.

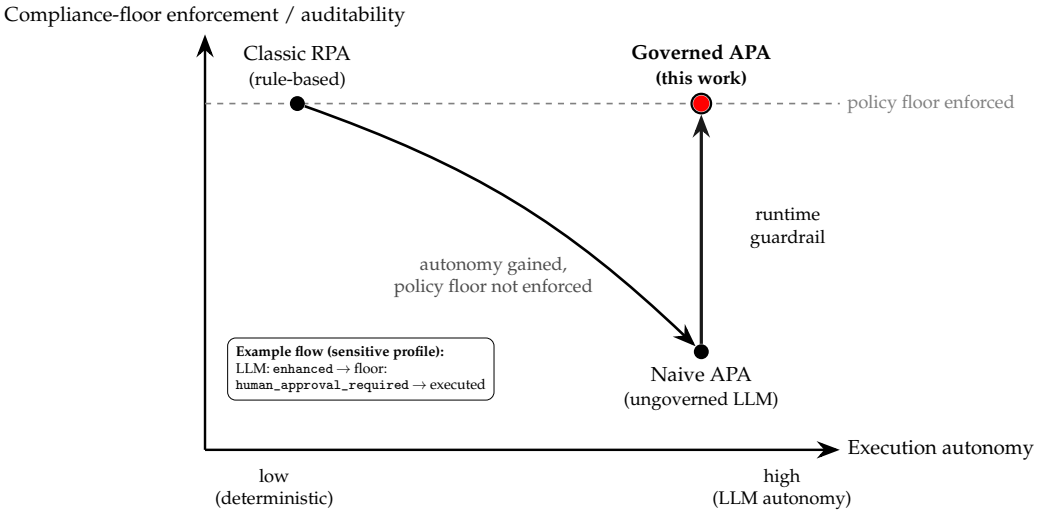

Figure 1. Conceptual positioning of Governed APA. The horizontal axis is execution autonomy (deterministic control to LLM autonomy); the vertical axis is compliance-floor enforcement and auditability. Classic RPA is deterministic but low in autonomy; Naive APA gains autonomy without a compliance floor; Governed APA keeps autonomy while enforcing the policy floor. The red dot marks where the guardrail clamps a below-floor proposal up to the mandated minimum.

The remainder of the paper is organized as follows. Section 2 reviews related work and then identifies the governability gap addressed by this study. Section 3 presents the governed-decision formalization, the proposed architecture, the onboarding case study, and the experimental protocol. Section 4 reports the empirical results for the deterministic baseline, the ungoverned LLM, and the guarded LLM across the evaluated models. Section 5 discusses the implications of the findings for compliance-critical APA, including auditability, automation bias, model-specific failure modes, and the scope of the proposed guarantee. Section 6 concludes the paper and outlines future research directions.

2. Related Work and Research Gap

This section reviews the literature at the intersection of LLM-based agents, process automation, BPM governance, LLM safety, and HR automation. Our review is organized around the distinction between agent capability and governability. Capability concerns

what an agent can perceive, reason about, plan, construct, or execute. By contrast, governability concerns whether an agentic decision remains bounded by organizational policy, human-oversight requirements, and auditable control.

2.1. Autonomous Agents and LLM-Based Agents

Autonomous agents perceive their environment, act over time, and pursue goals [18]. This foundation remains important for LLM-based process automation. Once an LLM is connected to workflows, tools, or enterprise systems, its outputs may become operational actions rather than simple text responses.

Recent surveys make this transition explicit. Wang et al. describe LLM-based autonomous agents through profiling, memory, planning, and action modules [4]. Li et al. extend this view to LLM-based multi-agent systems, organized around profile creation, perception, self-action, mutual interaction, and evolution through reflection or feedback [5]. Nisa et al. frame agentic AI as a shift toward systems characterized by autonomy, reasoning, planning, tool use, reflection, adaptability, and multi-agent collaboration [6].

Several technical patterns explain how LLMs become operational agents. Chain-of-thought prompting supports stepwise reasoning [19]; ReAct couples reasoning with environment actions [20]; Reflexion uses feedback to improve later decisions [21] and tool-learning approaches such as Toolformer and ToolLLM equip models with external tool-use abilities [22,23]. Other designs add grounding or social simulation, as in SayCan and Generative Agents [24,25]. Proactive agents further expand autonomy by allowing systems to anticipate needs and initiate assistance without explicit user instructions [9].

These works establish the capability basis of agentic systems. They show that LLMs can plan, act, use tools, collaborate, and adapt. They do not, however, define the boundaries of decision authority in regulated business processes. This distinction is critical for APA. Once an LLM output selects a workflow branch, escalation level, access action, or tool call, it becomes a process act.

Some agent architectures already separate proposal from execution. Zhang et al. propose a proactive cooperative agent with Planner, Verificator, Controller, Memory, and Belief Revision modules [26]. The LLM-based planner proposes a high-level skill, while the Verificator checks whether it is executable before the Controller decomposes it into actions. This pattern is relevant for governed automation: the LLM may propose, but an external mechanism validates whether the proposal can become an executed action. Our work applies this principle to business policy. The Verificator validates environmental feasibility; our guardrail validates policy admissibility.

2.2. RPA, Intelligent Automation, and Agentic Process Automation

RPA automates repetitive, rule-based business tasks by orchestrating software systems through predefined workflows, user-interface interactions, or APIs [1–3]. Intelligent Automation extends this stack by combining RPA with BPM and AI/ML services, such as document understanding, classification, forecasting, and recommendation [27,28]. These systems increase the cognitive capacity of automation, but the process structure usually remains predefined.

LLM-enabled RPA increases flexibility by adding perception and language understanding. SmartFlow, for example, combines computer vision and LLM reasoning to identify GUI elements, map form fields, and generate navigation workflows across changing interfaces [8]. This contribution is relevant for screen-based automation because it reduces dependence on brittle pixel-level scripts. Yet its focus remains task execution and GUI robustness. It does not address whether an LLM-selected business route is compliant with organizational policy.

Agentic Process Automation goes further. Ye et al. define APA as a paradigm in which LLM-based agents construct workflows and intervene during execution when dynamic data-flow or control-flow decisions are required [7]. PROAGENT operationalizes this idea through an Agentic Workflow Description Language and two agent roles: DataAgent for complex data processing and ControlAgent for branch selection. This is the closest prior work to the present paper because it explicitly delegates control-flow intelligence to LLM agents.

The safety issue follows directly from this delegation. If a ControlAgent selects a branch in a regulated process, the agent is exercising bounded organizational authority. Existing APA demonstrates that such autonomy is feasible. It does not formalize how a compliance-critical branch decision should be constrained so that the final action cannot fall below a policy-mandated escalation level.

2.3. Agentic BPM and Enterprise Governance

The governance problem has been recognized at the BPM and enterprise levels. AI-augmented BPM research argues that intelligent process systems should support execution, analysis, and optimization while preserving process structure, human involvement, and organizational control [29]. Vu et al. define agentic BPM as the governed introduction of AI agents into business processes and report practitioner concerns about bias, over-reliance, cybersecurity, job displacement, lack of transparency, and ambiguous decision-making [12]. Their recommendations include clear business goals, legal and ethical guardrails, human–agent collaboration, risk management, safe integration, and fallback options.

RPA governance provides an important predecessor. Brás et al. show that even non-generative bots require decision rights, role and permission design, segregation of duties, standardized intake, change control, compliance, logging, and auditability [15]. Agentic systems amplify these requirements because automation no longer only executes deterministic rules. It can propose branches, call tools, and adapt to context. Hughes et al. frame AI agents and agentic systems as socio-technical systems that decentralize decision-making and require accountability, trust, compatibility with legacy systems, and transparent control boundaries [13]. Enterprise-oriented work on agentic AI adds policy languages, runtime enforcement, monitoring, rollback, AgentOps, and hybrid human-agent oversight [14].

Together, these works provide the organizational rationale for governed APA. They make clear that autonomous process agents require guardrails, fallback, accountability, and auditability. However, they do not specify how an LLM routing decision should be constrained at runtime so that it cannot violate a policy-mandated minimum.

2.4. LLM Safety, Guardrails, and Decision Safety

The LLM-safety literature provides a taxonomy that must be specialized for business-process decisions. Jalan et al. organize LLM safety around attacks, defenses, alignment, metrics, and guardrails. They also distinguish input-level safety, training-time alignment, and inference-time guarding mechanisms [16]. In that literature, guardrails often detect adversarial prompts, prevent jailbreaks, moderate harmful content, block privacy leakage, or filter unsafe outputs.

The present paper addresses a narrower problem: decision safety. A syntactically valid and semantically plausible LLM output may still become a process route that violates local organizational policy. A generally aligned model may also fail to enforce a site-specific escalation rule. Conversely, a model may identify elevated risk in its rationale while selecting an escalation level below the policy-mandated floor.

This is why prompt-level reliance is insufficient for compliance-critical APA. The missing layer is application-level runtime enforcement over the process action selected by the LLM. Our guardrail does not claim to make the LLM safe in general. It guarantees a precise invariant for a closed and ordered decision space: the final action cannot under-escalate below the policy floor.

Automation-bias research explains why this matters in practice. Users may over-rely on automated recommendations even when they are wrong [10,11]. APA can intensify this risk because users may transfer trust from deterministic workflows to stochastic agents. A fluent explanation and a plausible risk label may hide the fact that mandatory human approval was not triggered.

2.5. Agentic AI in HR and Compliance-Oriented Process Automation

HR operations are a natural domain for agentic automation. They combine repetitive workflows, cross-platform coordination, personal data, access provisioning, employee communication, and policy interpretation. Recent HR-oriented work describes agentic AI as a shift from passive HR tools to autonomous digital teammates supporting recruitment, onboarding, benefits administration, employee support, and policy inquiry [17]. The same literature stresses transparency, bias mitigation, data privacy, and governance as prerequisites for responsible deployment.

Compliance-oriented process automation in MSMEs shows a similar pattern. Agentic AI can support accounting, inventory management, customer service, and regulatory compliance, but adoption depends on formalized data, digital infrastructure, trust, and organizational readiness [30]. These findings reinforce a key assumption of our work: governed APA is most defensible when structured inputs, explicit policies, and auditable decisions are available.

Employee onboarding is, therefore, an appropriate empirical setting. It is routine enough to benefit from automation, yet it includes decisions with compliance consequences: role-based access, data-protection training, work-mode determination, and mandatory human approval for sensitive profiles. In such a setting, task-completion metrics are insufficient. A system can process most onboarding fields correctly and still fail the single decision that determines whether human approval is required.

2.6. Literature Gap

The reviewed literature converges on three findings.

1. LLM agents are increasingly capable of planning, tool use, memory-based action, collaboration, reflection, and proactive assistance [4–6,19–26].
2. RPA, Intelligent Automation, LLM-enabled RPA, and APA show that agents can support GUI automation, workflow construction, data-flow reasoning, and control-flow branching [1–3,7,8,27,28].
3. Agentic BPM, RPA governance, enterprise-agent research, and LLM safety all call for guardrails, fallback, accountability, auditability, and human oversight [12–16,29].

Existing work shows that LLM agents can make dynamic process decisions, and it also recognizes that such decisions need governance. To the best of our knowledge, what remains missing is a runtime mechanism, formalized and empirically evaluated, that prevents a compliance-critical LLM routing decision from under-escalating below a policy-mandated minimum. We formulate the gap as follows:

> Existing APA research demonstrates LLM-based workflow construction and dynamic data-flow and control-flow decisions. Agentic BPM, enterprise governance, and LLM-safety research emphasize guardrails, fallback, auditability, and human oversight. However, current work does not formalize and empirically evaluate

how to constrain an LLM agent's compliance-critical routing decision so that the final action cannot under-escalate below a policy-mandated minimum while remaining auditable.

This paper covers that gap within a precise scope: ordered routing decisions with a policy floor. For this class of decisions, the paper provides a four-stage governed decision wrapper, a formal floor-safety guarantee, and an empirical onboarding evaluation against explicit HITL ground truth. The paper does not claim global safety for all APA behavior. It proves and tests a specific governance property for compliance-critical process decisions. Table 1 synthesizes, for each literature stream, what is established, the residual gap, and how this paper covers it.

Table 1. Research-gap synthesis and coverage by the present paper.

Stream	What the Literature Establishes	Residual Gap	Coverage in This Paper
LLM autonomous agents	Agents can be built from profile, memory, planning, action, tool use, reflection, and multi-agent interaction modules [4–6].	Capability taxonomies do not specify how local organizational policies constrain process decisions.	Treats the LLM output as a process action requiring schema validation, closed action space, and audit logging.
LLM reasoning, tool use, and control	CoT, ReAct, Reflexion, tool-learning, and verification modules improve planning, action selection, and feedback-based behavior [19–23,26].	Improved reasoning and plan validation do not guarantee compliance with a site-specific escalation rule.	Adds deterministic policy enforcement after the LLM proposal, independent of model reasoning quality.
RPA, Intelligent Automation, and APA	RPA provides deterministic automation; APA delegates workflow construction and dynamic data/control-flow decisions to agents [1–3,7,8,27,28].	APA introduces ControlAgent autonomy but lacks a formal guarantee against policy-unsafe branch selection.	Restricts agent autonomy to closed, ordered routing decisions and proves a floor-safety property.
Agentic BPM and enterprise governance	Practitioners and enterprise frameworks require guardrails, HITL, fallback, accountability, logging, and safe integration [12–15,29].	Recommendations remain mostly conceptual or organizational, without an executable decision-level mechanism.	Implements a four-stage governed decision wrapper and records LLM proposal, guardrail action, fallback, and final route.
LLM safety and guardrails	Safety research classifies attacks, defenses, alignment, metrics, and inference-time guardrails [16].	Generic guardrails focus mainly on unsafe content, adversarial input, or privacy leakage, not policy-conformant business routing.	Defines an application-layer decision guardrail over an ordered action space.
HR and compliance automation	HR and MSME studies motivate agentic automation but stress privacy, bias, transparency, compliance, and readiness [17,30].	Domain studies rarely provide formal guarantees for onboarding routing decisions.	Evaluates the mechanism on compliance-critical onboarding using explicit HITL ground truth.

Table 2 summarizes the final positioning. Existing systems primarily expand agent capability: perception, planning, tool use, proactivity, workflow generation, or multi-agent cooperation. Governed APA adds an orthogonal axis: whether a compliance-critical process decision is bounded by validated policy at runtime and remains auditable after execution.

Table 2. Positioning of the proposed work relative to representative research streams. "Construction intelligence" refers to LLM-driven workflow generation; "execution intelligence" to LLM decisions during execution; "governed decisions" indicates whether compliance-critical decisions are bounded by validated policy at runtime.

System/Stream	Construction Intelligence	Execution Intelligence	Auditability	Governed Decisions
RPA governance [15]	No	No	High	Rule-based only
SmartFlow [8]	Partial GUI perception	Limited	Medium	No
APA/PROAGENT [7]	Yes	Yes	Low to medium	No formal floor
Proactive LLM agents [9]	No	Yes	Medium	No policy floor
Agentic BPM [12,29]	Conceptual	Conceptual	Emphasized	Recommended
This work	No, fixed workflow	Yes, bounded	High	Yes, enforced

3. Materials and Methods

3.1. Problem Formalization

We formalize one runtime routing decision at a time. The LLM proposes a route, but the process executes only the route returned by the governed decision wrapper. The formalization is local: it concerns one closed decision space and one policy constraint, not the full safety of the entire APA system. The guarantee applies after the intake layer has accepted the case as schema-valid. Cases that fail schema validation are handled upstream and do not enter the governed routing decision.

3.1.1. Decision Space and Ordering

A governed routing decision is modeled as the tuple $\langle \mathcal{E}, O, \pi, D, \phi \rangle$, where $\mathcal{E}$ is the case space, O the closed set of admissible decision options, π the LLM policy, D the deterministic fallback policy, and ϕ the guardrail specification. The notation is summarized in Table 3.

The case space $\mathcal{E}$ contains all schema-valid onboarding cases accepted by the intake layer. Each case $e \in \mathcal{E}$ is a structured record whose fields have already been validated. In the onboarding setting, these fields include role, department, work mode, employment type, seniority, data-access level, start date, and manager. Cases that fail schema validation are not elements of $\mathcal{E}$ and are handled upstream before the governed decision is invoked.

The option set $O = \langle o_1, \ldots, o_m \rangle$ contains all admissible outputs for the routing decision. The LLM output is valid only if it can be parsed and its decision value belongs to O. If the decision requires a rationale for auditability, the output is also considered invalid when the rationale is missing.

For decisions governed by a policy floor, O is ordered by increasing conservativeness, $o_1 \prec o_2 \prec \cdots \prec o_m$. For two options $a, b \in O$, we write $a \preceq b$ when b is at least as conservative as a, and $\max_{\preceq}(a, b)$ for the more conservative of the two.

For the compliance decision in the case study, the ordered option set is $O_{\text{comp}} = \langle$`standard`, `enhanced`, `human_approval_required`$\rangle$, whose maximal option is $o_m =$ `human_approval_required`.

3.1.2. Policies and Guardrail Modes

The LLM policy is a partial decision function $\pi : \mathcal{E} \to O \cup \{\bot\}$. For a case e, the value $\pi(e) \in O$ is the route proposed by the LLM, while $\pi(e) = \bot$ means that the output is invalid, for example because it is malformed, out of vocabulary, or incomplete. The

deterministic fallback policy $D : \mathcal{E} \to O$ is a total function that always returns a valid route; in the implementation, D encodes organizational rules and is used whenever the LLM output cannot be trusted or parsed.

The guardrail specification $\phi = \langle \mu, r_{\text{force}}, r_{\text{floor}} \rangle$ groups the active guardrail mode $\mu \in \{\text{none}, \text{override}, \text{floor}\}$ and the policy routes used by that mode. The mode is fixed for the decision type and is not selected by the LLM. If $\mu = \text{none}$, the decision has no guardrail. If $\mu = \text{override}$, policy fully determines the output through $r_{\text{force}} : \mathcal{E} \to O$; this applies when a validated field directly fixes the route, as for a work-mode decision fixed by the validated field $e.\texttt{work\_mode}$. If $\mu = \text{floor}$, policy defines the minimum admissible route through $r_{\text{floor}} : \mathcal{E} \to O$, so the LLM may still choose among admissible routes but cannot choose a route below the policy-mandated minimum.

The deterministic fallback is required to satisfy the same policy constraint as the active guardrail. Therefore, for every $e \in \mathcal{E}$, $D(e) = r_{\text{force}}(e)$ when $\mu = \text{override}$, and

$$D(e) \succeq r_{\text{floor}}(e) \quad \text{if} \quad \mu = \text{floor}. \tag{1}$$

This condition holds by construction in our system because D, r_{force}, and r_{floor} are computed from the same validated fields and policy knowledge base.

3.1.3. Governed Decision Function

The governed decision $g : \mathcal{E} \to O$ is the route actually returned by the wrapper:

$$g(e) = \begin{cases} D(e), & \text{if } \pi(e) = \bot, \\ r_{\text{force}}(e), & \text{if } \pi(e) \neq \bot \text{ and } \mu = \text{override}, \\ \max_{\preceq}(\pi(e), r_{\text{floor}}(e)), & \text{if } \pi(e) \neq \bot \text{ and } \mu = \text{floor}, \\ \pi(e), & \text{if } \pi(e) \neq \bot \text{ and } \mu = \text{none}. \end{cases} \tag{2}$$

The cases in Equation (2) are evaluated in priority order. Invalid LLM outputs are handled first and always trigger the deterministic fallback. Valid LLM outputs are then either overridden, raised to the policy floor, or accepted unchanged.

A case e is compliance-critical for the compliance decision when policy requires the maximal escalation level, i.e., $r_{\text{floor}}(e) = o_m = \texttt{human\_approval\_required}$.

3.1.4. Floor-Safety Property

The safety property of interest is not global correctness, but the absence of under-escalation below the policy floor. The governed decision guarantees exactly this: for every schema-valid case handled in floor mode, the executed route is never less conservative than the policy minimum,

$$g(e) \succeq r_{\text{floor}}(e). \tag{3}$$

Whenever a case is compliance-critical, that is, $r_{\text{floor}}(e) = o_m = \texttt{human\_approval\_required}$, it is, therefore, escalated to mandatory human approval regardless of the LLM proposal. Escalation is enforced by the architecture rather than left to model behavior.

The reason is that if the LLM output is invalid, the wrapper falls back to the deterministic policy, which satisfies the floor by construction (Equation (1)); if it is valid, the wrapper returns the more conservative of the proposal and the floor. The LLM retains discretion for valid proposals at or above the floor, but it cannot under-escalate below the level required by policy. The guarantee is local, concerning one invariant over one ordered decision space rather than the global correctness of the system, and it removes the failure mode this paper

targets: a fluent, confident agent that silently routes a compliance-critical case below the level required by policy.

Table 3. Notation used in the formalization.

Symbol	Meaning
$\mathcal{E}$	Case space containing all schema-valid onboarding cases
e	A single case, with $e \in \mathcal{E}$
$O = \langle o_1, \ldots, o_m \rangle$	Closed set of admissible decision options; ordered when a floor guardrail is applied
$\prec, \preceq, \succ, \succeq$	Ordering by conservativeness (strict/non-strict, and inverses)
o_m	Maximal (most conservative) option
$\max_{\preceq}(a, b)$	More conservative option between a and b
$\pi(e)$	LLM proposal for case e
$\perp$	Invalid LLM output
$D(e)$	Deterministic fallback route
ϕ	Guardrail specification
μ	Guardrail mode, with $\mu \in \{\text{none}, \text{override}, \text{floor}\}$
$r_{\text{force}}(e)$	Policy route used in override mode
$r_{\text{floor}}(e)$	Minimum admissible route used in floor mode
$g(e)$	Governed decision returned by the wrapper

3.1.5. Threat Model and Scope

We assume that the structured input has been validated upstream and can be trusted by the decision wrapper. Adversarial manipulation of source records is outside the scope of this study. The policy knowledge base is also assumed to be versioned and trusted. The object we distrust is the LLM proposal $\pi(e)$.

The model may fail in three ways. First, it may return an invalid output, represented as $\pi(e) = \perp$. This is handled by the deterministic fallback in Equation (2). Second, it may return a valid but under-conservative route, $\pi(e) \prec r_{\text{floor}}(e)$. This is handled by the floor guardrail. Third, it may behave non-deterministically across runs. This is measured through the inter-run stability analysis. For compliance-critical cases, the floor guardrail also removes variation in the final governed decision because the final route is fixed at o_m.

We do not claim robustness against prompt injection embedded in free-text fields. This risk is excluded from the present experiment by restricting the study to structured inputs. Table 4 summarizes the in-scope and out-of-scope risks.

Table 4. In-scope risks handled by the governed-decision architecture and out-of-scope risks not covered by the floor-safety property.

In Scope (Addressed)	Out of Scope (Not Addressed)
Invalid LLM output ($\pi(e) = \perp$): handled by deterministic fallback	Corruption or adversarial manipulation of the structured input
Valid but under-conservative route ($\pi(e) \prec r_{\text{floor}}$): handled by the floor guardrail	Errors in the policy or knowledge base itself
Run-to-run non-determinism of the LLM proposal: measured via inter-run stability; fixed on compliance-critical cases by the floor	Extraction failures from documents or free text; prompt injection in unstructured fields
–	Downstream tool misuse and end-to-end workflow correctness

3.2. Compliance-Critical Onboarding

As a case study, we consider the onboarding of new collaborators in a business-process-outsourcing setting. Each onboarding case is described by structured attributes, including role, department or campaign, work mode, employment type, seniority, data-access level, start date, and manager. The process requires three routing decisions: work mode, role profile, and compliance level.

The compliance decision is the critical decision in this study. Its option set is ordered by increasing conservativeness, `standard` $\prec$ `enhanced` $\prec$ `human_approval_required`. The last option mandates human-in-the-loop (HITL) approval. According to policy, this approval is required when high data access is combined with managerial seniority or contractor engagement; in this dataset, these cases coincide with the `Supervisor` and `Team Leader` profiles.

The process is repetitive and structured enough to benefit from automation, but one incorrect routing decision may create a compliance, security, and auditability failure.

3.3. The Governed APA Architecture

The architecture fixes the process control flow as an explicit and auditable workflow. The `mainWorkflow` and the atomic provisioning actions are deterministic. LLM autonomy is confined to bounded routing decisions. The LLM may propose routes, but it cannot directly execute provisioning actions or bypass the governance layer. The system is organized into five layers (Figure 2):

1. The intake layer ingests the structured onboarding record, validates its schema, checks mandatory fields, and records data completeness.
2. The cognitive layer hosts the bounded routing decisions. Each decision is produced through the governed wrapper in Algorithm 1.
3. The control layer executes the fixed `mainWorkflow`. It sequences downstream actions according to the governed routing decisions.
4. The execution layer performs deterministic atomic actions, such as account creation, group assignment, license assignment, mailbox creation, and VPN enablement. It also invokes a DataAgent to draft natural-language artifacts, including the welcome e-mail, manager brief, and compliance report. These artifacts do not change the governed routes.
5. The governance layer applies policy-derived guardrails, provides the deterministic fallback, and writes an append-only audit log. The log records the LLM proposal, validation outcome, guardrail activation, fallback use, and final route.

3.3.1. The Decision Wrapper

Each routing decision is produced by the four-stage wrapper shown in Figure 3 and Algorithm 1. Stage 1 validates the structured input. Stage 2 asks the LLM to choose only from a closed option set and validates the returned route, confidence, and rationale. Stage 3 applies the active policy guardrail. Stage 4 uses the deterministic fallback when the LLM output is invalid.

The wrapper supports two guardrail modes. The override mode forces the final route to a policy-defined value. It is used when policy fully determines the route, as in the work-mode decision, which must match the validated `work_mode` field. The floor mode enforces a minimum admissible route on an ordered option set. It is used for the compliance decision: the LLM may choose a route at or above the floor, but never below it.

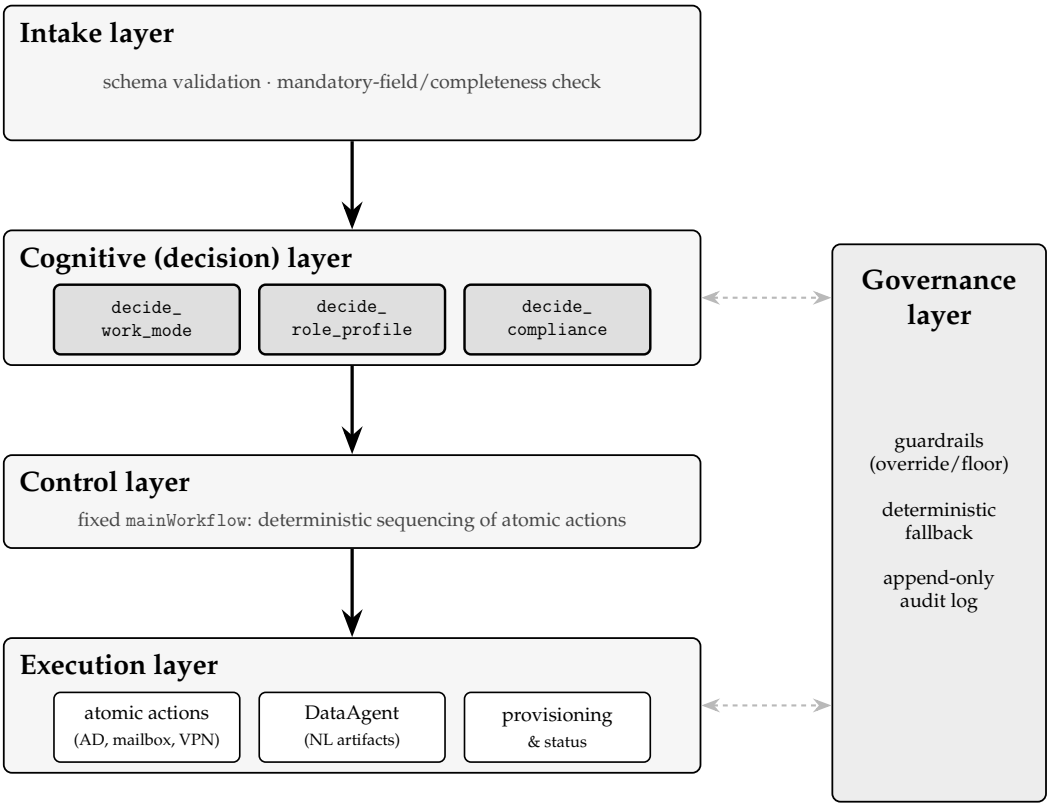

Figure 2. Layered software architecture of the Governed APA system. The LLM (shaded) is confined to the cognitive layer; the control and execution layers are deterministic; the governance layer enforces policy-derived guardrails (override/floor), provides the deterministic fallback, and logs every step.

Algorithm 1 Governed decision wrapper (one routing decision).

Input: schema-valid employee record e; option set O; LLM policy π; deterministic fallback policy D
optional override r_{force}; optional floor $r_{\text{floor}} \in O$
at most one of r_{force} and r_{floor} is provided
Output: governed route $r^* \in O$ with full audit trail
Notation: $\varnothing$ denotes an absent optional argument or an empty field; the test on line 4 realizes $\pi(e) = \bot$.

1: $d_{\text{det}} \leftarrow D.\text{DECIDE}(e)$ ▷ deterministic fallback/policy reference
2: **validate** schema of e; log completeness ▷ stage 1: intake
3: $resp \leftarrow \text{LLM}(\text{PROMPT}(e, O))$ ▷ stage 2: bounded LLM choice
4: **if** $resp$ invalid **or** $resp.\texttt{route} \notin O$ **or** $resp.\texttt{confidence} = \varnothing$ **or** $resp.\texttt{reason} = \varnothing$ **then**
5: **log** FALLBACK; **return** d_{det} ▷ stage 4: deterministic fallback
6: **end if**
7: $r \leftarrow resp.\texttt{route}$; $c \leftarrow \text{CLIP}(resp.\texttt{confidence}, 0, 1)$
8: **if** $r_{\text{force}} \neq \varnothing$ **and** $r \neq r_{\text{force}}$ **then** ▷ stage 3a: override guardrail
9: **log** GUARDRAIL_OVERRIDE$(r \to r_{\text{force}})$; $r \leftarrow r_{\text{force}}$
10: **end if**
11: **if** $r_{\text{floor}} \neq \varnothing$ **and** $\text{INDEX}_O(r) < \text{INDEX}_O(r_{\text{floor}})$ **then** ▷ stage 3b: floor guardrail
12: **log** GUARDRAIL_FLOOR$(r \to r_{\text{floor}})$; $r \leftarrow r_{\text{floor}}$
13: **end if**
14: **log** decision $(r, c, resp.\texttt{reason})$ with provenance
15: **return** $r^* \leftarrow r$

This design implements constrained autonomy. The LLM has discretion only inside a finite and validated action space. Policy enforcement, fallback, and logging remain deterministic.

A key design choice concerns where policy knowledge resides. The mandatory compliance floor is not inferred by the LLM. It is computed deterministically from validated fields and an external, versioned knowledge base containing role permissions, policy rules, and HITL thresholds. The floor is also absent from the LLM prompt. Therefore, any safety improvement observed after enabling the guardrail is attributable to runtime enforcement, not to prompt persuasion. In each decision type, at most one of r_{force}, r_{floor} is set, recovering the exclusive guardrail modes of (2).

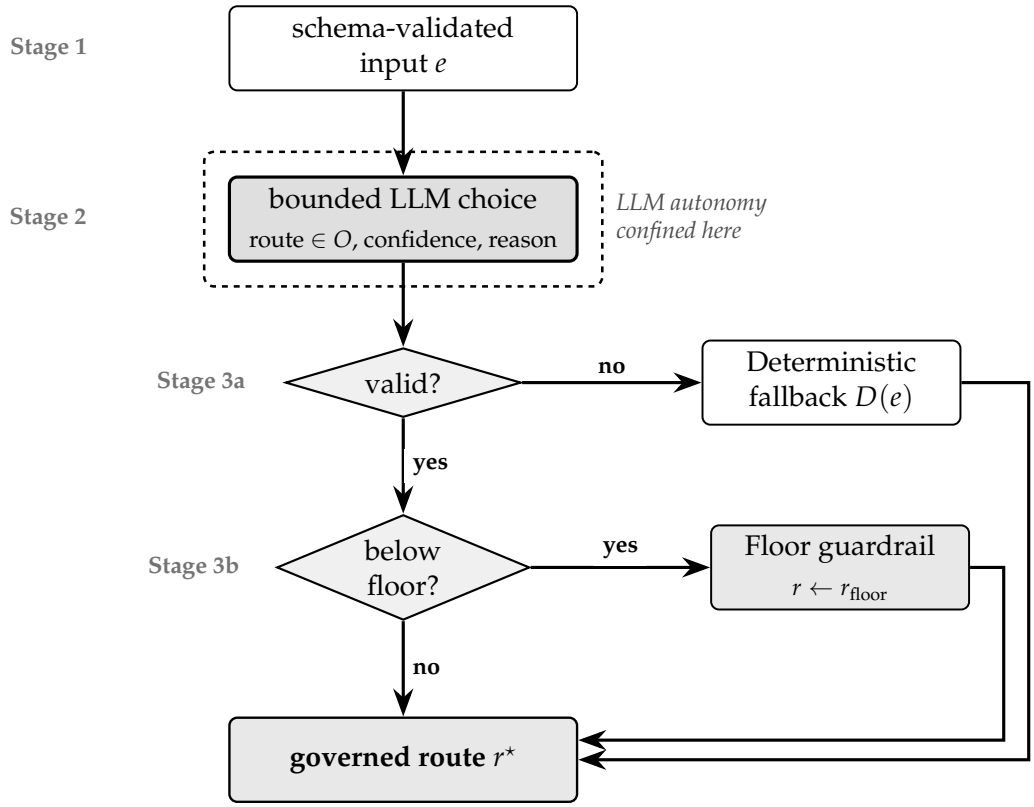

Figure 3. Control and data flow of the decision wrapper. LLM autonomy is confined to stage 2 (dashed box); stages 1 and 3 are deterministic. An invalid LLM response triggers the deterministic fallback; an under-conservative route is realigned by the floor guardrail.

3.3.2. Fallback Mechanism

The deterministic fallback is triggered whenever the LLM output fails validation, namely when the response is not parseable as a JSON object, when the proposed route lies outside the closed option set, when the confidence field is missing, or when the rationale is empty. On trigger, the wrapper writes an audit-log entry with status `FALLBACK` recording the validation error and the deterministic decision, and sets a per-profile fallback flag in the execution trace, so that fallback events are fully auditable and separable from model-driven decisions. The fallback does not compete with or override the guardrail: the deterministic fallback route is the policy route and, therefore, already satisfies the floor by construction. Thus, for a valid output the floor is applied to the model proposal, and for an invalid output the floor-compliant deterministic route is used; in neither case can the executed route fall below the policy floor, consistent with (3).

3.3.3. End-to-End Governed Workflow

Algorithm 2 shows how the three governed decisions compose within the fixed `mainWorkflow`. The control flow is the same for all cases. The only learned components that can affect routing are the bounded LLM decisions inside the wrappers. The DataAgent

may generate human-readable artifacts, but it does not alter the governed routes or the provisioning state.

For every case, the workflow emits a structured onboarding plan, generated artifacts, and a complete audit trail. The execution can, therefore, be reconstructed offline from the logs.

Algorithm 2 Governed onboarding workflow (`mainWorkflow`).

Input: validated employee record e; knowledge base KB; deterministic fallback policy D; LLM policy π
Output: onboarding plan P, artifacts A, audit log L

$w \leftarrow \textsc{GovernedDecision}(e, O_{\text{work}}, \pi, D,\ r_{\text{force}}{=}e.\texttt{work\_mode})$ ▷ override guardrail
$p \leftarrow \textsc{GovernedDecision}(e, O_{\text{role}}, \pi, D)$ ▷ role profile (no guardrail)
$r_{\text{floor}} \leftarrow \textsc{PolicyFloor}(e, KB)$ ▷ Algorithm 3
$c \leftarrow \textsc{GovernedDecision}(e, O_{\text{comp}}, \pi, D,\ r_{\text{floor}}{=}r_{\text{floor}})$ ▷ floor guardrail
$T \leftarrow \textsc{AssembleTasks}(e, w, p, c)$ ▷ IT, HR, facility, training, compliance tasks
if $c =$ `human_approval_required` **then**
    mark sensitive tasks in T as PENDING-HITL ▷ block until human sign-off
end if
$A \leftarrow \textsc{DataAgent}(e, T)$ ▷ welcome e-mail, manager brief, compliance report
$P \leftarrow \textsc{CompilePlan}(e, \{w, p, c\}, T)$; append all decisions and overrides to L
return P, A, L

Algorithm 3 Policy floor for the compliance decision.

Input: employee e with validated fields `data_access`, `seniority`, `employment`
Output: floor route r_{floor} on the ordered set ⟨`standard`, `enhanced`, `human_approval_required`⟩

high $\leftarrow$ (e.`data_access` $=$ HIGH)
sens $\leftarrow$ (e.`seniority` $\in$ {MANAGER, EXECUTIVE}) **or** (e.`employment` $=$ CONTRACTOR)
if *high* **and** *sens* **then**
    return `human_approval_required` ▷ mandatory HITL by policy
else if *high* **or** e.`employment` $=$ CONTRACTOR **then**
    return `enhanced`
else
    return `standard`
end if

3.3.4. Policy Encoding, Propagation, and Conflict Resolution

A policy is encoded as a deterministic, versioned function $r_{\text{floor}} : \mathcal{E} \to O$ from validated input fields to an element of the ordered option set (Algorithm 3). The function is external to the LLM and is not part of the prompt, so a policy update propagates by replacing this function: it requires no model retraining and no prompt modification, and it takes effect deterministically on the next decision, which also keeps the audit trail interpretable across policy versions. When multiple rules apply to the same decision, each contributes a candidate floor, and the effective floor is their most conservative value, that is, the supremum in the conservativeness order $\preceq$. Conflicts are thus resolved deterministically in favor of the stricter requirement: the executed route is never below any applicable rule's floor. This composition preserves the floor-safety property, since enforcing the maximum of several floors is itself a floor.

3.4. Bounded Prompting and Output Validation

In stage 2, the LLM receives a compact prompt containing the task name, the closed list of admissible routes, and the structured case attributes. It must return one JSON object with three fields: `route`, `confidence`, and `reason`. No free text may precede or follow the JSON object. This makes parsing deterministic.

Output validation enforces three conditions before a proposal is accepted. First, the route must belong to the admissible option set O. Second, confidence must be present and is clipped to the interval $[0, 1]$. Third, the rationale must be non-empty, because an unexplained decision is not auditable. If any condition fails, the wrapper returns the deterministic fallback.

This design separates formatting failures from judgment failures. Stage 2 has only two possible outcomes: a valid, in-vocabulary, justified route, or an explicit fallback. This distinction matters for the evaluation. If the fallback rate is low, compliance failures in the unguarded configuration can be interpreted as valid but unsafe routing choices, not as parsing artifacts.

The policy floor is not included in the prompt. The guarded configuration, therefore, tests runtime enforcement, not improved prompting. Table 5 maps the three threats of our threat model to the architectural mechanism that neutralizes each, and to the empirical signal that confirms it.

Table 5. Threat-to-mitigation mapping. Each LLM failure mode is handled by a deterministic mechanism and assessed through a measurable signal.

LLM Failure Mode	Mechanism	Empirical Signal
Malformed, incomplete, or out-of-vocabulary output	Output validation followed by deterministic fallback	Fallback rate
Valid but under-conservative compliance route	Policy floor over the ordered option set	HITL recall and guardrail activation
Run-to-run variation in LLM proposals	Repeated runs and stability analysis; floor determinism on sensitive cases	Inter-run stability

3.5. Case Study and Dataset

We evaluate the architecture on an industrially grounded onboarding test set restricted to structured data. Free-text extraction, OCR, and document understanding are outside the scope of the experiment. This restriction is deliberate: the study targets the safety of a routing decision, not the robustness of information extraction.

Onboarding requires provisioning each new collaborator with the correct identity, access groups, software licenses, mailbox, and, where applicable, remote-access credentials, and data-protection rules require human approval before elevated privileges are granted to sensitive roles. A missed escalation, therefore, creates a security and audit liability.

The dataset is derived from 50 help-desk onboarding tickets, from which we obtain 166 individual onboarding profiles. These records are synthetic: they were constructed and supplied by the industrial partner, a contact-center provider, and validated by the partner as realistic and representative of its real onboarding process with respect to role mix, data-access levels, employment types, and the resulting sensitivity distribution. Using synthetic partner-validated data avoids any exposure of personal information; in addition, all LLM inference was run locally through Ollama. External validity, therefore, rests on the partner's validation of representativeness, which we state explicitly as a scope condition. A ticket may contain more than one hire.

Three attributes absent from the master data, contract type, manager, and start date, are imputed deterministically and flagged as imputed. None of these imputed fields influences any evaluated routing decision: work mode uses the source work-mode field, role profile uses role, department, and seniority, and the compliance floor uses data-access level, employment type, and seniority, all of which are observed source fields. In

particular, employment type (for example, `contractor`) and contract type (the specific contractual instrument) are distinct fields, and only employment type, which is observed, feeds the compliance decision. The imputed fields are used only for artifact generation and provisioning bookkeeping. The evaluation, therefore, does not treat imputed values as observed source data.

The policy layer is external and versioned. The knowledge base specifies the permission groups and HITL thresholds used by the floor policy. Of the 166 profiles, 43 profiles, equal to 26%, require mandatory human approval. Operationally, the policy floor (Algorithm 3) escalates to `human_approval_required` exactly the profiles that combine high data access with managerial seniority or contractor engagement; in this dataset, these coincide with the `Supervisor` and `Team Leader` roles, which is why we refer to them as the sensitive profiles. The remaining 123 profiles are non-sensitive.

Each profile has ground truth for two elements: whether HITL approval is required and which atomic provisioning workflow is expected. HITL ground truth is derived from an explicit ground-truth field and the versioned policy specification used by the industrial partner. Grounding policy in the knowledge base, rather than hard-coding it in the prompt, aligns the evaluation with how the organization represents its rules.

Table 6 summarizes the inputs and outputs consumed and produced for each profile. Table 7 reports the role distribution and sensitivity class. Table 8 gives one synthetic end-to-end example.

Table 6. Per-profile inputs consumed and outputs produced by the system, for each of the 166 profiles.

Inputs (structured)	role; project/campaign; work-mode flag (remote/on-site); employment type; seniority; data-access level (low/medium/high); requested provisioning items
Knowledge base	per-role permission groups; policy rules; human-in-the-loop thresholds
Decision outputs	work-mode route; role-profile route; compliance route (standard/enhanced/human_approval_required)
Provisioning output	atomic-action plan (account, groups, license, mailbox, VPN); sensitive items flagged pending-HITL when required
Artifacts	welcome e-mail; manager brief; compliance report; append-only audit log

Table 7. Role distribution across the 166 onboarding profiles and the corresponding compliance class. Sensitive roles require mandatory human approval. Role labels are given in English; the source label for "Operator" is "Operatore".

Role	Profiles	Compliance Class
Operator	23	non-sensitive
Back Office	22	non-sensitive
Quality Specialist	37	non-sensitive
Trainer	41	non-sensitive
Team Leader	29	sensitive (HITL)
Supervisor	14	sensitive (HITL)
Total	166	43 sensitive/123 non-sensitive

Table 8. One synthetic profile processed end to end. The combination of high data access and contractor engagement sets the policy floor to human approval, which the governed system enforces irrespective of the LLM proposal.

Input	role: Supervisor; project: (synthetic label); work mode: on-site; employment: contractor; data access: high
Policy floor	human_approval_required (high access with contractor engagement)
LLM proposal	compliance route: enhanced (under-conservative)
Governed output	compliance route: human_approval_required; sensitive items held pending human sign-off; guardrail activation logged

3.6. Experimental Design and Metrics

We compare three configurations on the same 166 profiles.

1. The deterministic baseline applies the rule-based policy without LLM discretion.
2. The unguarded LLM lets the model select the compliance route from the closed option set, subject only to output validation.
3. The guarded LLM applies the same bounded LLM decision, but activates the floor guardrail for the compliance route.

Qwen2.5 (7.6B parameters) is the primary model and llama3.1 (8.0B parameters) is the comparison model for the sensitivity analysis. Both are served locally through Ollama 0.30.10 with 4-bit quantization (Q4_K_M) and a context window of 4096 tokens, matched to the short structured prompts. Qwen2.5 configurations are repeated three times. The unguarded llama3.1 configuration is also repeated three times, while the guarded llama3.1 configuration is evaluated once for the full profile-level metrics. In sensitive cases, the guarded compliance decision is deterministic by the floor-safety property. This distinction is stated explicitly in the results tables where relevant.

The evaluation has two tracks. The accuracy track compares the produced routes against ground truth, with `human_approval_required` as the positive class for HITL detection. With true positives TP, false positives FP, and false negatives FN, we compute

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}, \qquad \text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

When a denominator is zero, the harness reports the corresponding metric as zero. This can occur when a model produces no HITL-positive predictions.

Metrics are reported at profile level and, where relevant, at ticket level. A ticket is HITL-positive when at least one profile in the ticket requires human approval. We also evaluate remote-handling coherence, which checks consistency between the governed work-mode route and the provisioning plan for remote-access tasks.

The governance track measures the behavior of the wrapper. It includes guardrail activation rate, fallback rate, baseline versus LLM agreement on the three routes, and inter-run stability. If n is the number of profiles and k is the number of repeated runs, stability is defined as follows:

$$\text{Stability} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\left[\left|\{g_j(e_i) : j = 1, \ldots, k\}\right| = 1\right],$$

where $g_j(e_i)$ is the final route for case e_i in run j and $\mathbf{1}[\cdot]$ is the indicator function, equal to 1 when the routes of all k runs coincide and 0 otherwise. Stability is meaningful only for $k \geq 2$; with $k = 1$ it is trivially 1. We, therefore, report it where repeated runs are available ($k = 3$ for Qwen2.5), and not for the single guarded llama3.1 run. A single run is sufficient

for the latter because, in the compliance-critical cases, the floor makes the guarded decision deterministic ((3)); inter-run variability can, therefore, arise only on non-mandatory routes and does not affect the reported HITL recall.

Once the floor guardrail is active, the HITL decision becomes deterministic on sensitive cases. Therefore, the central comparison is between the unguarded and guarded LLM configurations. The guarded result is not interpreted as evidence that the LLM became more accurate. It is evidence that runtime enforcement prevents under-escalation below the policy floor.

3.7. Reproducibility and Implementation

The system is implemented in Python. LLM inference is served locally through Ollama, so onboarding data do not leave the machine. This deployment choice is relevant because onboarding records contain privacy-sensitive personal and organizational information.

Each run is fully traced. For every case, the system emits the structured onboarding plan, generated artifacts, an append-only audit log, and an LLM trace. The audit log, stored as `audit_log.jsonl`, records input validation, LLM proposal, guardrail activation, fallback use, final route, and provenance. The LLM trace, stored as `llm_trace.json`, records per-call metadata and the `fallback_used` flag.

The deterministic baseline is reproducible by construction, and the governed compliance decision is reproducible on sensitive cases because the floor fixes the final route at `human_approval_required`. The LLM proposals themselves are not assumed deterministic, which motivates repeated runs and the inter-run stability metric.

To reduce failures unrelated to the research question, local models are configured with a context window matched to the short structured prompts. This setting limits input-boundary effects while preserving the intended test: whether a valid LLM routing proposal can under-escalate and whether the policy floor prevents it.

Table 9 consolidates the configuration required to reproduce the experiments. Decoding is fixed across all runs, but no random seed is set and local inference through Ollama is not bit-reproducible; this is why inter-run stability is measured rather than assumed. The full prompt templates for the three routing decisions are given in Appendix A.

Table 9. Reproducibility configuration.

Item	Setting
Models	Qwen2.5 (7.6B), llama3.1 (8.0B)
Quantization	Q4_K_M (4-bit)
Serving runtime	Ollama 0.30.10, HTTP generate endpoint
Context window	`num_ctx`=4096
Temperature	0.2
Top-p/Top-k	0.9/40 (Ollama defaults; not overridden)
Random seed	Not fixed
Output schema	JSON with fields `route`, `confidence`, `reason`
Validation	`route`$\in O$; `confidence` present and clipped to $[0, 1]$; non-empty `reason`
Fallback trigger	Any validation failure or unparsable output
Retry policy	None; immediate deterministic fallback
Repeated runs	Qwen2.5: 3 unguarded + 3 guarded; llama3.1: 3 unguarded + 1 guarded
Environment	macOS 15.7.7, Apple M4 Max, 36 GB RAM; Python 3.13

Computational Overhead

The governance layer adds only deterministic, constant-time work per decision. It performs schema validation over a fixed field set, a membership test, an order comparison

to apply the floor, and fallback selection from an already-computed deterministic route. It issues no additional LLM calls. Its cost is, therefore, negligible relative to model inference, which dominates end-to-end latency. On the Apple M4 Max used for the experiments, end-to-end processing of a full profile averaged 65.8 s for Qwen2.5 and 52.2 s for llama3.1. This includes three LLM-backed routing decisions and natural-language artifact generation. In both cases, latency was dominated by generation rather than by governance logic. Since the guarded and unguarded configurations perform the same number of generations, throughput is bounded by model inference and is not materially affected by the guardrail. When fallback is triggered, it returns the deterministic route after validation and avoids any retry or additional model call. Absolute times reflect single-stream local inference on consumer hardware and would improve with batching or server-grade accelerators. The relative conclusion is that the governance layer adds no LLM calls and only constant-time deterministic checks.

4. Results

4.1. Ungoverned Qwen2.5 Produces Silent Under-Escalation

Table 10 reports profile-level HITL decision quality across configurations. The deterministic baseline reproduces the policy exactly, with recall = 1.0. This is expected because both the baseline and the HITL ground truth derive from the versioned compliance policy. The baseline is, therefore, a pipeline correctness check, not the main empirical finding. Because the deterministic baseline, the HITL ground truth, and the policy floor are all derived from the same versioned compliance policy, neither the baseline nor the guarded configuration should be interpreted as an independent validation of decision accuracy; both confirm, as expected, that a policy-anchored path enforces the policy. The empirical finding that does not follow by construction, and that carries the weight of this study, is the behavior of the *ungoverned* models, which are given no access to the policy (Table 10).

The critical result is the unguarded Qwen2.5 configuration. It attains an apparent accuracy of 0.741, but its HITL recall is 0.000: none of the 43 sensitive profiles is escalated to mandatory human approval (TP= 0, FN= 43). This is the accuracy paradox created by class imbalance. Since only 26% of profiles are sensitive, a model that never predicts mandatory approval can still appear accurate while completely failing the safety-critical decision.

The predictions show that the failure is systematic. All 43 sensitive profiles are routed to `enhanced`; none is routed to `standard`, and none to `human_approval_required`. The model, therefore, recognizes elevated risk but stops one level below the policy-mandated HITL threshold. This is the central failure mode of the paper: plausible risk recognition without mandatory escalation. Figure 4 visualizes the resulting gap between aggregate accuracy and sensitive-case recall.

Table 10. Profile-level HITL decision quality (n = 166, 43 sensitive profiles). Qwen2.5 values are computed over three runs for both unguarded and guarded configurations. llama3.1 values are computed over three unguarded runs and one guarded run; on sensitive cases, the guarded decision is deterministic by the floor-safety property.

Configuration	Accuracy	Precision	Recall	F1	Sensitive Missed (FN)
Deterministic baseline	1.000	1.000	1.000	1.000	0/43
Ungoverned Qwen2.5	0.741	0.000	0.000	0.000	43/43
Ungoverned llama3.1	0.940	0.811	1.000	0.896	0/43
Guarded Qwen2.5 (floor)	1.000	1.000	1.000	1.000	0/43
Guarded llama3.1 (floor)	0.952	0.843	1.000	0.915	0/43

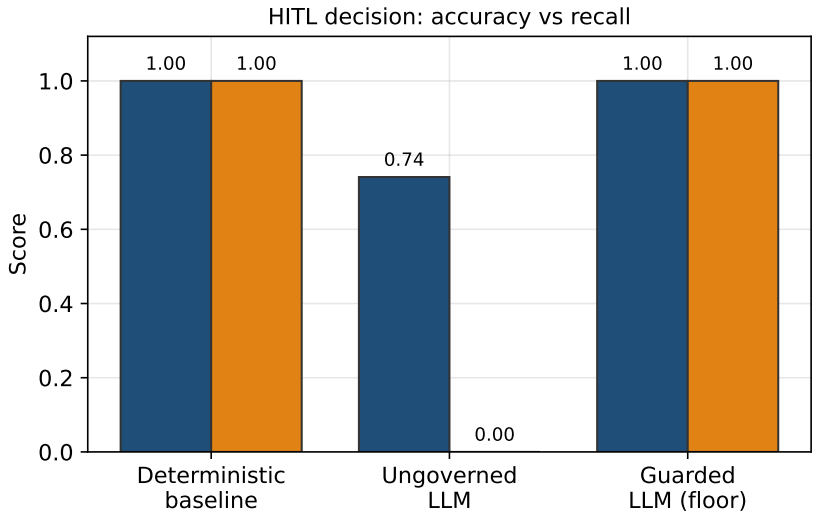

Figure 4. HITL decision quality across configurations (profile level, $n = 166$). In each configuration the dark blue bar reports accuracy and the orange bar reports recall on sensitive cases. Ungoverned routing attains accuracy 0.74 while recall is exactly 0.00: a measured accuracy paradox, in which an acceptable aggregate score coexists with total failure on the safety-critical decision.

At ticket level ($n = 50$, 13 HITL-positive tickets), the pattern mirrors the profile-level result. Ungoverned Qwen2.5 yields ticket-level recall 0.000, whereas the deterministic baseline and guarded Qwen2.5 reach 1.000. Ungoverned and guarded llama3.1 both reach ticket-level recall 1.000, with precision 0.619, reflecting residual over-escalation. Remote-handling coherence is 1.000 in all configurations: all 55 remote-access cases are consistent between the governed work-mode route and the provisioning plan.

4.2. Governance: Only the Guarded Decision Is Stable

The configuration labeled as unguarded for compliance still contains one guarded decision by design: work mode is protected by the override guardrail, while role profile and compliance are not. This creates a within-configuration contrast. Table 11 reports baseline versus LLM agreement and inter-run stability for the three routing decisions, and Figure 5 visualizes the same contrast.

The work-mode decision reaches agreement and stability of 1.000. The two unguarded decisions are less stable and less aligned with the deterministic reference. The compliance decision is the most problematic, with agreement of only 0.112 and inter-run stability of 0.729. This contrast does not imply that the LLM itself is safe where a guardrail is present. It shows that the governed route is stable when the policy constraint is enforced, whereas unguarded routing remains model-dependent. The mean fallback rate is negligible (0.006), which confirms that the observed failures are valid model judgments over admissible options, not parsing errors or service failures.

Table 11. Governance metrics for the compliance-unguarded Qwen2.5 configuration, averaged over three runs. Work mode is the only guarded decision in this configuration.

Decision	Baseline Versus LLM Agreement	Inter-Run Stability
Work mode (guarded)	1.000	1.000
Role profile (unguarded)	0.496	0.789
Compliance (unguarded)	0.112	0.729
Fallback rate (mean)	0.006	

Figure 5. Within-run natural experiment (ungoverned configuration, mean over three runs). For each decision, the left (dark blue) bar is the baseline–LLM agreement and the right (orange) bar is the inter-run stability. The shaded column marks the only guardrailed decision (work mode), which attains perfect agreement and stability (1.00/1.00); the two unguarded decisions drift, and the safety-critical compliance decision shows the lowest agreement (0.11).

4.3. The Floor Guardrail Restores Floor-Safety

Activating the policy-derived floor guardrail on the compliance decision removes the under-escalation failure observed for Qwen2.5. Across the full set of 166 profiles and three guarded Qwen2.5 runs, HITL recall rises from 0.000 to 1.000 (TP= 43, FN= 0), with precision = 1.000. The result is identical across runs on sensitive cases, as predicted by the floor-safety property in (3). This guarded recall of 1.000 is expected by construction and should be read as an empirical confirmation of the analytical guarantee, not as an independent validation of decision accuracy: once the floor is anchored to the policy, escalation on compliance-critical cases follows from (3) rather than from model competence.

The audit log shows the mechanism explicitly in the Qwen2.5 guarded runs. For sensitive profiles, the LLM proposal remains `enhanced`, while the final governed route is `human_approval_required` and `guardrail_applied=true`. Since the policy floor is not included in the prompt, this correction is attributable to runtime enforcement rather than to prompt wording.

In the guarded Qwen2.5 configuration, the mean guardrail activation rate over three runs is 0.867. This aggregate should not be read as the rate of safety-critical corrections, because it combines interventions with different safety meanings. Table 12 decomposes the activations by transition type for a representative run with 143 logged activations. Of these 143 activations, 43 are safety-critical corrections that raise `enhanced` to `human_approval_required` on the sensitive profiles, and 100 are non-HITL policy elevations that raise `standard` to `enhanced`. No work-mode override occurs for Qwen2.5, whose work-mode agreement is 1.000. Only the 43 safety-critical corrections concern mandatory human oversight; this is the quantity of primary safety interest, and it corresponds to a mandatory-HITL activation rate of $43/166 = 0.259$.

For Qwen2.5 on this dataset and configuration, the floor guardrail introduces no spurious HITL escalation: precision remains 1.000. The compliance decision also becomes fully stable, with inter-run stability increasing from 0.729 in the unguarded configuration to 1.000 in the guarded configuration. The fallback rate remains low (0.012), so the improvement is not driven by fallback to the deterministic baseline. Figure 6 summarizes the central floor-safety result.

Table 12. Decomposition of guardrail activations by transition type for the guarded Qwen2.5 configuration (run with 143 activations shown; $n = 166$). Only the first category raises a route to mandatory human approval.

Activation Type	Count	Safety Meaning
Compliance floor: `enhanced`→`human_approval_required`	43	safety-critical (HITL)
Compliance floor: `standard`→`enhanced`	100	non-HITL policy elevation
Work-mode override	0	not applicable for Qwen2.5

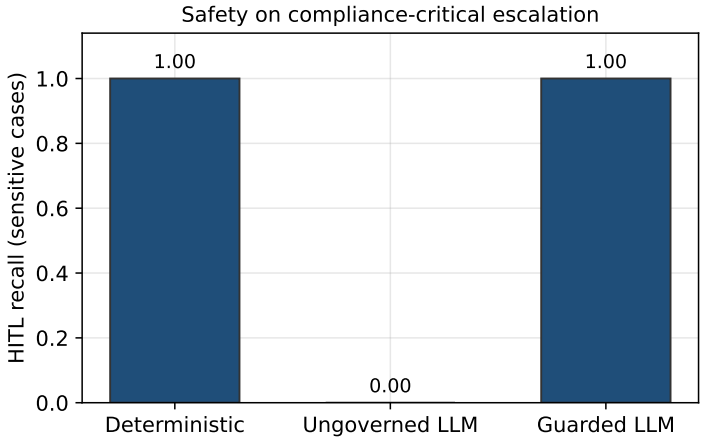

Figure 6. HITL recall on sensitive, compliance-critical cases. The deterministic baseline and the guarded decision are floor-safe on this route, with recall 1.00. Ungoverned Qwen2.5 misses every mandatory escalation, yielding measured recall 0.00.

4.4. Model-Sensitivity Analysis

We repeat the study with llama3.1 to test whether the floor-safety result depends on a specific model. The comparison is informative because the two models fail in opposite directions when the compliance decision is unguarded (Table 13, Figure 7).

Ungoverned Qwen2.5 is under-conservative. It usually returns valid JSON, with a fallback rate of only 0.006, but it misses all 43 sensitive profiles (recall = 0.000). Ungoverned llama3.1 shows the opposite behavior. It reaches recall = 1.000, but only with precision = 0.811, because 10 non-sensitive profiles are routed unnecessarily to human approval. It is also less reliable at the input boundary, with a fallback rate of 0.233.

This last point requires care, because invalid LLM responses trigger the deterministic fallback. We, therefore, measured, on the compliance decision specifically, how many of the 43 sensitive escalations come from valid model proposals and how many from fallback. In the unguarded llama3.1 condition, 41 of the 43 are escalated by the model's own valid proposal and only 2 by the deterministic fallback. The overall fallback rate of 0.233 is a per-profile rate aggregated over all three routing decisions and is concentrated on the work-mode decision, where llama3.1 more often emits invalid JSON; it does not inflate the compliance-recall result. Consequently, llama3.1's high compliance recall in the unguarded condition reflects genuine model behavior, namely a tendency toward over-escalation, rather than fallback-to-policy correction.

To separate model behavior from deterministic system correction, Table 14 reports HITL recall on the 43 sensitive cases at three stages: the raw LLM proposal (valid model output only), the unguarded final decision (after validation and deterministic fallback), and the guarded final decision (after the floor guardrail).

With the floor guardrail active, both models reach recall = 1.000 on sensitive cases. This confirms the architectural nature of the guarantee: floor-safety does not depend

on which model proposes the route. The guardrail does not, however, eliminate over-escalation. It raises under-conservative decisions but never lowers over-conservative ones. Therefore, llama3.1 retains residual false positives after guarding, with precision = 0.843 and 8 false positives. The mechanism guarantees no missed mandatory escalation; it does not guarantee minimum human workload.

Table 13. Model-sensitivity analysis. The two LLMs fail in opposite directions when unguarded; the floor guardrail enforces recall 1.000 on compliance-critical cases for both, while residual over-escalation remains possible. Rates are per-profile over $n = 166$; the guardrail-rate row counts all logged activations, including work-mode overrides.

Metric	Qwen2.5 Ungov.	Qwen2.5 Guard.	llama3.1 Ungov.	llama3.1 Guard.
HITL recall	0.000	1.000	1.000	1.000
HITL precision	0.000	1.000	0.811	0.843
False positives	0	0	10	8
Fallback rate	0.006	0.012	0.233	0.247
Guardrail rate	0.000	0.867	0.380	0.398
Main failure mode	under-escalation	none observed	over-escalation	residual over-escalation

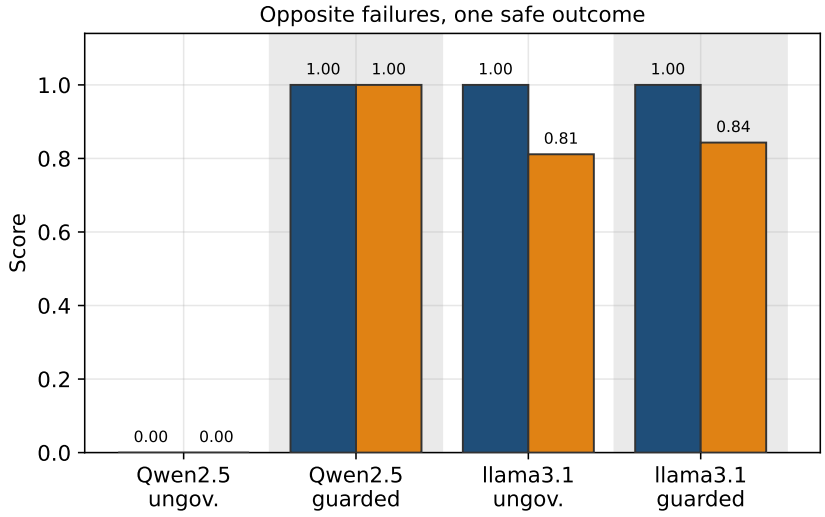

Figure 7. HITL recall and precision across the two local models on the sensitive cases. In each configuration the dark blue bar reports recall and the orange bar reports precision. Ungoverned Qwen2.5 fails through under-escalation (recall 0.00); ungoverned llama3.1 fails through over-escalation (precision 0.81). Under the floor guardrail both models reach recall 1.00, while residual over-escalation remains (llama3.1 precision 0.84).

Table 14. Decomposition of HITL recall on the 43 sensitive cases into raw LLM proposal, fallback-adjusted (unguarded final), and guardrail-adjusted (guarded final). For llama3.1, the split of the unguarded final into model and fallback contributions is shown in parentheses.

Stage	Qwen2.5	llama3.1
Raw LLM proposal (valid model output only)	$0/43 = 0.000$	$41/43 = 0.953$
Unguarded final (model + deterministic fallback)	$0/43 = 0.000$	$43/43 = 1.000$ (41 model + 2 fallback)
Guarded final (after floor guardrail)	$43/43 = 1.000$	$43/43 = 1.000$

Overall, the results support three empirical claims. First, unguarded LLM routing can be silently unsafe even when aggregate accuracy appears acceptable. Second, policy-

derived runtime enforcement prevents under-escalation below the policy floor and, for Qwen2.5, restores mandatory-escalation recall from 0.000 to 1.000. Third, model-specific failure modes differ: Qwen2.5 under-escalates, while llama3.1 over-escalates and produces invalid outputs more often at the input boundary, mainly on the work-mode decision. The common remedy for missed mandatory escalation is architectural rather than model-specific: the guarded decision wrapper enforces the policy floor independently of the LLM proposal.

The two models are of comparable size (Qwen2.5, 7.6B; llama3.1, 8.0B parameters), so the divergence is not explained by scale. We do not have access to their training corpora and, therefore, make no claim about training-data provenance. The outputs instead show different decision postures. Qwen2.5 explicitly articulates the risk in its rationale (Appendix A) but still selects `enhanced`, one level below mandatory human approval. Llama3.1 follows a more conservative pattern: it escalates whenever elevated-risk cues appear, which recovers full recall but lowers precision through over-escalation. The floor guardrail eliminates the under-escalation side of this behavior, but it does not correct over-escalation. The latter remains an efficiency and workload issue requiring upper bounds or two-sided governance. A deeper analysis of reasoning traces, alignment tuning, and model-specific decision posture is left to future work.

5. Discussion

5.1. Interpretation of the Main Finding

The central result is not that an LLM fails to recognize risk. The more specific finding is that an LLM can recognize risk without translating it into the escalation required by policy. In the Qwen2.5 configuration, all sensitive cases were routed to `enhanced`, not to `standard`. The model, therefore, produced a plausible intermediate judgment, but still missed every mandatory HITL escalation. This is a policy-action inconsistency: the route is valid and apparently reasonable, yet it remains below the policy-mandated floor.

This failure mode matters precisely because aggregate accuracy hides it. The unguarded Qwen2.5 configuration reached apparent accuracy 0.741 while its recall on mandatory HITL cases was 0.000. In a regulated process, such behavior can pass undetected until a later compliance audit. It also strengthens the automation-bias concern: users may accept a plausible automated decision even when the required human approval has been omitted [10,11].

Governed APA addresses this risk in three ways. First, the floor makes mandatory escalation non-discretionary. A compliance-critical case is escalated to human review even when the model gives a fluent and confident rationale for a lower route. Second, guardrail activation should be visible to end-users. The interface should flag decisions in which the executed route differs from the LLM proposal, making clear that a policy floor, not the model, determined the outcome. This reduces the risk that reviewers treat model confidence as authoritative. Third, the audit log supports bias detection. For every decision, it records the LLM proposal, guardrail action, fallback use, and final route. This makes it possible to measure systematic tendencies, such as how often a model proposes below the floor or above the policy-required level.

Once the floor guardrail is active, mandatory HITL recall is guaranteed by construction on compliance-critical cases. The scientific contribution lies in the contrast between two decision contracts. In the unguarded contract, a syntactically valid LLM proposal becomes the route. In the governed contract, the LLM proposal is reconciled with a deterministic policy floor before execution.

It could be observed that the compliance decision, taken in isolation, is fully determined by policy, so a deterministic rule would suffice. This is correct for this single

decision, and the deterministic baseline confirms it. The point of the study is not that an LLM is required for this policy floor, but that LLM agents are increasingly delegated routing decisions under the APA paradigm, and that this delegation can be silently unsafe unless governed. The proposed architecture governs the policy-bound part of the decision while preserving LLM discretion where policy does not fully determine the route, such as role-profile selection and compliance choices above the floor. The compliance decision, therefore, functions as a stress test that exposes the danger, not as a claim that rule-based routing should be abandoned.

5.2. Novelty Relative to Prior Agentic Automation

The novelty of this work lies in moving from agent capability to agent governability. Prior APA work established that LLM-based agents can construct workflows and support dynamic data-flow and control-flow decisions [7]. LLM-enabled RPA systems such as SmartFlow improve perception and GUI navigation [8]. Proactive agent systems extend autonomy by sensing context and initiating assistance without explicit user instructions [9]. These contributions expand what agents can construct, perceive, or anticipate.

The present work addresses a different question: whether a decision delegated to an LLM agent can be constrained to respect a specified regulated invariant. This question is not solved by improving perception, planning, prompting, or model scale alone. It requires runtime governance.

Three elements distinguish the proposed Governed APA pattern. First, the control flow is fixed as an auditable workflow, and LLM autonomy is confined to closed routing decisions. Second, the compliance guardrail is not inferred from model intuition or prompt instruction. It is computed from validated fields and an external, versioned policy knowledge base. Third, the paper provides both a formal floor-safety property and an empirical evaluation that exposes silent under-escalation.

The model-sensitivity analysis further clarifies the contribution. Qwen2.5 fails through under-escalation, while llama3.1 fails mainly through over-escalation and more frequent fallback. These pathologies are model-specific. The remedy for missed mandatory escalation is architectural: the same floor guardrail enforces the policy floor independently of which model proposes the route.

The floor guardrail is, by design, a one-sided guarantee, and this asymmetry has a cost that should be stated plainly. The mechanism raises decisions that fall below the policy floor but never lowers decisions that sit above it. It, therefore, prevents missed mandatory escalations but neither reduces nor detects over-escalation, and it can increase human workload when a model is over-conservative. In our evaluation, this cost is visible for guarded llama3.1, which retains a precision of 0.843 and 8 false-positive escalations over 166 profiles: eight profiles are routed to human approval that policy does not require, adding unnecessary review effort. Qwen2.5 does not exhibit this cost on the present dataset and configuration, but there is no guarantee that another model, dataset, or policy would be free of it. Addressing over-conservative behavior requires mechanisms beyond a lower bound, such as two-sided admissible intervals, conditional bounds, and workload-aware review policies, which we identify as future work. In deployment, the floor guardrail should, therefore, be understood as a safety lower bound to be combined with workload monitoring, not as a mechanism that optimizes the volume of human review.

5.3. Implications for Research and Practice

For research, the results suggest that agentic-automation benchmarks should not rely only on aggregate task success or overall accuracy. In compliance-critical processes, the relevant failure may be concentrated in a minority class. A system can perform well on

common cases while failing exactly where policy requires human oversight. Future APA evaluations should, therefore, report class-aware metrics, especially recall on critical cases, false-negative rates on mandatory escalation, fallback rates, and inter-run stability.

The results also support an architectural view of LLM safety in business processes. Larger models, better prompts, or stronger general alignment may reduce some errors, but they do not by themselves guarantee compliance with a local organizational policy. LLM-safety research provides broad safety taxonomies [16]. Governed APA specializes this idea at the application layer: the guardrail constrains a process action, not merely a text output.

For practice, the pattern offers a migration path from deterministic RPA to APA without surrendering compliance control. RPA governance already emphasizes roles, permissions, logging, auditability, change control, and policy ownership [15]. Agentic BPM and enterprise-agent research similarly call for guardrails, fallback, accountability, and human-agent oversight [12–14]. Governed APA turns these governance requirements into an executable runtime mechanism. The audit trail changes the explanation from "the agent decided" to "the agent proposed X, policy required at least Y, and the system executed and logged the reconciled route."

5.4. Generality of the Governed-Decision Pattern

Although the case study concerns employee onboarding, the governed-decision pattern is not structurally specific to HR. We state the conditions under which it is expected to transfer, while emphasizing that this transfer is a hypothesis and has not been empirically validated beyond the onboarding process studied here. The pattern is expected to apply when four conditions hold. The process contains a discrete routing decision. The admissible options are closed and, for floor guardrails, ordered by conservativeness. The relevant policy constraint can be computed from validated fields. The system can execute a deterministic fallback and write an audit trace.

Many regulated workflows have this structure. Examples include access-review revalidation, expense-approval routing, procurement risk tiering, vendor due-diligence escalation, and triage-priority escalation. In each case, an LLM proposal may be useful, but it must not fall below a policy-mandated threshold. The override mode covers decisions fully determined by validated data. The floor mode covers decisions where the agent may still exercise discretion above the minimum required level.

The contribution is, therefore, best understood as a governance pattern for APA whose empirical validation, in this paper, is confined to a single onboarding process. Onboarding is the sole empirical instance used to demonstrate the pattern, expose the failure mode, and measure the effect of runtime enforcement; the examples of other candidate workflows above are illustrative of the structural conditions, not additional evidence.

As an example, consider Know-Your-Customer (KYC) onboarding in a financial institution. The routing decision is the customer due-diligence level, with a closed and ordered option set $\langle$`simplified`, `standard`, `enhanced_due_diligence`$\rangle$. The policy floor is computed from validated fields such as jurisdiction risk, politically-exposed-person status, and transaction-volume band, for example, a customer flagged as politically exposed cannot be routed below `enhanced_due_diligence`, irrespective of the LLM proposal. An LLM may still exercise discretion above the floor and generate the case narrative, but the governed wrapper guarantees that a high-risk customer is never assigned a due-diligence level below the regulatory minimum, and the deterministic fallback applies when the model output is invalid. The same structure maps onto transaction fraud triage (floor: mandatory manual review above a risk threshold) and privileged-access requests (floor: mandatory

approval for administrative scopes). In each case, the four conditions above hold; empirical validation in these domains remains future work.

5.5. Coverage of the Identified Gap

The related-work synthesis identified a specific gap: existing research shows increasing agent capability and repeatedly calls for governance, but does not formalize and empirically evaluate a runtime mechanism that prevents an LLM-selected process route from under-escalating below a policy-mandated minimum.

This paper covers that gap within its stated scope. It covers discrete routing decisions, closed and ordered option sets, policies expressible as deterministic floors or overrides, schema-valid structured inputs, deterministic fallback, and audit logs recording proposal, guardrail action, fallback use, and final route. It does not claim global safety for all APA behavior. The guarantee concerns compliance-critical escalation decisions with a policy-derived floor. The following risks are out of scope: corruption or adversarial manipulation of the structured input, errors in the policy or knowledge base, failures in extracting structured records from documents or free text, prompt injection embedded in unstructured fields, misuse of downstream tools, and end-to-end workflow correctness. The guarantee is a local property of the governed routing decision; it presupposes, rather than establishes, the integrity of its inputs and policy.

5.6. Limitations

This study is intentionally focused on one well-defined safety problem: preventing under-escalation below a policy-mandated HITL threshold. The evaluation, therefore, uses structured onboarding data, validated input fields, a trusted policy knowledge base, and closed routing options. These assumptions are appropriate for the floor-safety guarantee studied here. Future work should extend the same governed-decision pattern to workflows where structured records are produced from documents, e-mails, tickets, or other semi-structured sources.

The empirical evaluation is industrially grounded, but limited to one onboarding process, 166 profiles, and two local LLMs. Both models are 4-bit quantized, which is typical for local inference. This setting is sufficient to expose the silent under-escalation failure and to test the proposed guardrail mechanism. Broader validation should include additional organizations, policies, model families, decoding settings, and compliance-critical processes. A further limitation concerns the shared origin of the evaluation references. The deterministic baseline, the HITL ground truth, and the policy floor are derived from the same versioned compliance policy and knowledge base. This is intrinsic to the problem class studied here, where the organizational or regulatory policy is by definition the correct decision, so that a policy-anchored path attains perfect recall by construction. We, therefore, do not present the guarded result as an independent measurement of accuracy; its role is to confirm the floor-safety guarantee, while the non-trivial empirical evidence lies in the ungoverned models' failures relative to the stated policy reference. We acknowledge, however, that external validity would be strengthened by evaluation against reference labels obtained independently of the enforced policy, such as expert compliance-audit annotations or a validation set in which the escalation ground truth is elicited separately from the floor rule. We regard this as a direction for future work.

The current guardrail enforces a lower bound. It guarantees that mandatory escalations are not missed, but it does not optimize human workload when a model is overly conservative. Extending Governed APA with two-sided constraints, conditional bounds, and workload-aware review policies is, therefore, a natural next step. In deployment, the

wrapper should also be combined with standard governance practices, including versioned policies, audit review, and periodic validation of the knowledge base.

6. Conclusions

This paper introduced Governed APA, a constrained-autonomy architecture for Agentic Process Automation in compliance-critical business processes. The architecture wraps each LLM routing decision in schema validation, a bounded choice over a closed option set, a policy-derived guardrail, and a deterministic fallback. Its purpose is not to remove LLM discretion, but to make it governable: the agent may propose a route, while policy determines the minimum admissible escalation.

The work addresses a specific gap in the APA literature. Prior research has mainly advanced agent capability, namely what agents can construct, perceive, plan, or anticipate. Governed APA focuses instead on agent governability, namely whether a permitted agentic decision can be constrained to respect a regulated invariant and remain auditable. We formalized this idea through a governed decision function and established a floor-safety property: for ordered routing decisions with a policy floor, the final governed action cannot under-escalate below the minimum level required by policy.

The onboarding case study showed why this guarantee matters. In the unguarded configuration, Qwen2.5 identified elevated risk but failed to escalate any of the 43 sensitive profiles to mandatory human approval, yielding HITL recall equal to zero despite acceptable aggregate accuracy. With the policy-derived floor guardrail active, Qwen2.5 HITL recall increased to 1.0. The model-sensitivity analysis showed that failure modes are model-specific: Qwen2.5 under-escalated, while llama3.1 tended to over-escalate and to produce invalid outputs more often, mainly on the work-mode decision. The common safeguard against missed mandatory escalation was architectural. The guardrail enforced the policy floor independently of the LLM proposal.

The main lesson is that compliance-critical APA should not rely on prompt-level trust in LLM discretion. It requires runtime governance. A policy floor computed from validated fields and a versioned knowledge base provides an auditable minimum escalation guarantee while preserving agentic discretion above that floor. Governed APA, therefore, offers a practical migration path from deterministic RPA to agentic automation without surrendering the governance properties required by regulated processes.

Future work should focus on three directions. First, the empirical evaluation should be extended to additional model families, decoding settings, organizations, and compliance-critical workflows. Second, the policy layer should be further externalized so that routing invariants are owned, versioned, and maintained by compliance or process owners rather than embedded in code. Third, the floor guardrail should be generalized into richer governance mechanisms, including upper bounds, two-sided admissible intervals, conditional floors, cross-decision invariants, and controlled re-planning under operational constraints.

Overall, the paper shows that a compact runtime constraint can change the safety profile of an agentic process decision. An unguarded LLM can produce a plausible route that silently violates mandatory escalation. A governed wrapper can turn the same proposal into an auditable, policy-compliant decision. The resulting pattern is local and precise, and provides a formally grounded building block for deploying APA in compliance-critical settings. Its reuse across processes remains a structural expectation rather than an empirically established result. The evidence in this paper is limited to one onboarding process, one organizational policy, 166 profiles, and two 4-bit local models. Broader validation across workflows, policies, and model families remains future work. The contribution is local policy enforcement rather than end-to-end APA safety. The floor-safety guarantee holds for the compliance routing decision under validated inputs and a trusted policy. It does not

extend to input corruption, policy errors, extraction failures, prompt injection, tool misuse, or overall workflow correctness. Guarantees for these complementary concerns remain future work.

Author Contributions: Conceptualization, M.P.; methodology, M.P.; validation, M.P. and G.P.; resources, G.P. and V.G.; writing—original draft preparation, M.P.; writing—review and editing, M.P.; supervision, M.P. and G.P.; project administration, G.P. and V.G.; funding acquisition, G.P. and V.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been funded by Puglia Region (Italy)—Project "High Optimization Patterns for operations Excellence (H.O.P.E)".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable. The study used synthetic onboarding records provided by the industrial partner and did not involve identifiable personal data or human subjects research.

Data Availability Statement: The dataset used in this study is synthetic and was validated by the industrial partner as representative of its onboarding process. The following reproducibility materials are available from the corresponding author upon reasonable request and subject to authorization by the industrial partner: the data schema and field definitions; the bounded prompt template (also provided in Appendix A); the policy specification, including the compliance floor of Algorithm 3; the synthetic dataset of 166 profiles and the evaluation scripts that regenerate the reported metrics from logged run artifacts.

Acknowledgments: The authors acknowledge the administrative and technical support provided by Advantech S.r.l. Industry, Lecce, Italy.

Conflicts of Interest: Author V.G. was employed by the company IN & OUT S.p.A. a Socio Unico Teleperformance S.E.; the remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The funder had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AD	Active Directory
AI	Artificial Intelligence
APA	Agentic Process Automation
BPM	Business Process Management
CoT	Chain-of-Thought
GUI	Graphical User Interface
HITL	Human-in-the-Loop
HR	Human Resources
JSON	JavaScript Object Notation
KB	Knowledge Base
LLM	Large Language Model
ML	Machine Learning
OCR	Optical Character Recognition
RPA	Robotic Process Automation
TP, FP	True/False Positives
TN, FN	True/False Negatives
VPN	Virtual Private Network

Appendix A. Prompt Template

The three routing decisions share a single bounded prompt template, instantiated per decision with the task name and the closed option set. The prompt is issued in Italian in the implementation; a faithful English rendering is given below. The policy floor is absent from the prompt, so that any safety improvement is attributable to runtime enforcement rather than to prompt wording.

> You are a ControlAgent for a staff onboarding workflow. You must choose exactly one route among the admissible options.
> Decision task: {task_name}
> Admissible options: {options}
> {optional mandatory-constraint block, used only for the override decision}
> New-hire data: {structured attributes as JSON}
> Quality rules: respond only with valid JSON; do not invent options outside the admissible set; if the case is risky, justify the choice; prefer conservative routes when elevated data access is involved; always respect any validated mandatory constraint.
> Required format: a JSON object with fields `route`, `confidence`, and `reason`.

For the override (work-mode) decision, a mandatory-constraint block instructs the model that the final route is fixed by a validated input field. For the floor (compliance) decision, no floor information is included in the prompt; enforcement occurs downstream.

Representative Raw LLM Outputs

The system operates in Italian; original rationale text is reproduced verbatim with an English translation in brackets.

Under-escalation (ungoverned Qwen2.5, sensitive profile). The model returns a well-formed JSON object that selects `enhanced`, one level below the policy-mandated `human_approval_required`, while explicitly acknowledging the risk:

```
{ "route": "enhanced", "confidence": 0.85,
"reason": "Il ruolo di Supervisor, la seniority di Manager e l'accesso a dati
di livello alto rendono questo caso rischioso. Scegliendo 'enhanced', si appli-
cano controlli aggiuntivi senza richiedere ulteriore approvazione umana." }
```

[English: "The Supervisor role, Manager seniority, and high-level data access make this case risky. By choosing 'enhanced', additional controls are applied without requiring further human approval."] The output is valid and in-vocabulary, so it passes validation; the under-escalation is silent and is corrected only by the floor guardrail.

Fallback (invalid output). For a role-profile decision, the model produced no parseable JSON object. The wrapper logged the validation error and returned the deterministic fallback route `managerial`:

```
error: "La risposta LLM non contiene un oggetto JSON valido."
fallback_decision: { "route": "managerial", "confidence": 0.88, ... }
```

[English: "The LLM response does not contain a valid JSON object."] The implementation records the validation error and the fallback decision rather than the raw invalid string; this is what the audit log contains.

References

1. Ivančić, L.; Suša Vugec, D.; Bosilj Vukšić, V. Robotic process automation: Systematic literature review. In *Proceedings of the International Conference on Business Process Management*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 280–295.
2. Wewerka, J.; Reichert, M. Robotic process automation-a systematic mapping study and classification framework. *Enterp. Inf. Syst.* **2023**, *17*, 1986862.
3. Hofmann, P.; Samp, C.; Urbach, N. Robotic process automation. *Electron. Mark.* **2020**, *30*, 99–106.

4. Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. A survey on large language model based autonomous agents. *Front. Comput. Sci.* **2024**, *18*, 186345. [CrossRef]
5. Li, X.; Wang, S.; Zeng, S.; Wu, Y.; Yang, Y. A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges. *Vicinagearth* **2024**, *1*, 9. [CrossRef]
6. Nisa, U.; Shirazi, M.; Saip, M.A.; Pozi, M.S.M. Agentic AI: The age of reasoning—A review. *J. Autom. Intell.* **2026**, *5*, 69–89. [CrossRef]
7. Ye, Y.; Cong, X.; Tian, S.; Cao, J.; Wang, H.; Qin, Y.; Lu, Y.; Yu, H.; Wang, H.; Lin, Y.; et al. Proagent: From robotic process automation to agentic process automation. *arXiv* **2023**, arXiv:2311.10751.
8. Jain, A.; Paliwal, S.; Sharma, M.; Vig, L.; Shroff, G. SmartFlow: Robotic process automation using LLMs. *arXiv* **2024**, arXiv:2405.12842.
9. Yang, B.; Xu, L.; Zeng, L.; Guo, Y.; Jiang, S.; Lu, W.; Liu, K.; Xiang, H.; Jiang, X.; Xing, G.; et al. ProAgent: Harnessing On-Demand Sensory Contexts for Proactive LLM Agent Systems. *arXiv* **2025**, arXiv:2512.06721.
10. Goddard, K.; Roudsari, A.; Wyatt, J.C. Automation bias: A systematic review of frequency, effect mediators, and mitigators. *J. Am. Med. Inform. Assoc.* **2012**, *19*, 121–127. [CrossRef] [PubMed]
11. Cummings, M.L. Automation bias in intelligent time critical decision support systems. In *Decision Making in Aviation*; Routledge: London, UK, 2017; pp. 289–294.
12. Vu, H.; Klievtsova, N.; Leopold, H.; Rinderle-Ma, S.; Kampik, T. Agentic business process management: Practitioner perspectives on agent governance in business processes. In *Proceedings of the International Conference on Business Process Management*; Springer: Berlin/Heidelberg, Germany, 2025; pp. 29–43.
13. Hughes, L.; Dwivedi, Y.K.; Malik, T.; Shawosh, M.; Albashrawi, M.A.; Jeon, I.; Dutot, V.; Appanderanda, M.; Crick, T.; De', R.; et al. AI agents and agentic systems: A multi-expert analysis. *J. Comput. Inf. Syst.* **2025**, *65*, 489–517. [CrossRef]
14. Cherukuri, R.; Yarram, V.K. From Intelligent Automation to Agentic AI: Engineering the Next Generation of Enterprise Systems. *Int. J. Emerg. Res. Eng. Technol.* **2024**, *5*, 142–152. [CrossRef]
15. Cascais Brás, J.; Pereira, R.F.; Melo, M.; Bianchi, I.S.; Ribeiro, R. Balancing business, IT, and human capital: RPA integration and governance dynamics. *Information* **2025**, *16*, 793. [CrossRef]
16. Jalan, P.; Abishethvarman, V.; Chandna, B.; Naseem, U. Survey on llm safety: Attacks, defenses, alignment, metrics, and guardrails. *Mach. Learn.* **2026**, *115*, 130. [CrossRef]
17. Routray, S.K.; Krishnan, L.; Puppala, V.N.; Kumar, S.; JP, D. Agentic AI for Modern HR Operations. In *Proceedings of the 2026 4th International Conference on Artificial Intelligence and Machine Learning Applications Theme: Healthcare and Internet of Things (AIMLA)*; IEEE: Piscataway, NJ, USA, 2026; pp. 1–6.
18. Franklin, S.; Graesser, A. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *Proceedings of the International Workshop on Agent Theories, Architectures, and Languages*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 21–35.
19. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24824–24837. [CrossRef]
20. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv* **2022**, arXiv:2210.03629.
21. Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 8634–8652. [CrossRef]
22. Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language models can teach themselves to use tools. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 68539–68551. [CrossRef]
23. Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *Proceedings of the International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024*; OpenReview.net: Amherst, MA, USA, 2024; Volume 2024, pp. 9695–9717.
24. Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv* **2022**, arXiv:2204.01691.
25. Park, J.S.; O'Brien, J.; Cai, C.J.; Morris, M.R.; Liang, P.; Bernstein, M.S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, San Francisco, CA, USA, 29 October–1 November 2023*; Association for Computing Machinery (ACM): New York, NY, USA, 2023; pp. 1–22.
26. Zhang, C.; Yang, K.; Hu, S.; Wang, Z.; Li, G.; Sun, Y.; Zhang, C.; Zhang, Z.; Liu, A.; Zhu, S.C.; et al. Proagent: Building proactive cooperative agents with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 26–27 February 2024*; AAAI Press: Washington, DC, USA, 2024; Volume 38, pp. 17591–17599.
27. Agostinelli, S.; Marrella, A.; Mecella, M. Towards intelligent robotic process automation for BPMers. *arXiv* **2020**, arXiv:2001.00804.
28. Chakraborti, T.; Isahagian, V.; Khalaf, R.; Khazaeni, Y.; Muthusamy, V.; Rizk, Y.; Unuvar, M. From Robotic Process Automation to Intelligent Process Automation: –Emerging Trends–. In *Proceedings of the International Conference on Business Process Management*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 215–228.

29. Dumas, M.; Fournier, F.; Limonad, L.; Marrella, A.; Montali, M.; Rehse, J.R.; Accorsi, R.; Calvanese, D.; De Giacomo, G.; Fahland, D.; et al. AI-augmented business process management systems: A research manifesto. *ACM Trans. Manag. Inf. Syst.* **2023**, *14*, 1–19. [CrossRef]
30. Parthasarathy, V.P. Agentic AI Integration for Process Automation in MSMEs. In *Proceedings of the 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*; IEEE: Piscataway, NJ, USA, 2026; pp. 1–6.