Computer Fraud and Security
ISSN (online): 1873-7056

Architecting an Autonomous Compliance Infrastructure: An Agentic AI Framework for Legal-Critical Tax Systems

Abdul Basit Iqbal
Independent Researcher, USA

Abstract
Enterprise software infrastructure operates under stringent regulatory, security, and operational governance constraints. The manual translation of natural-language statutes into executable application logic introduces measurable error rates and systemic latency, with legacy update cycles requiring three to sixteen months for full integration. This article proposes a formalized Autonomous Compliance Infrastructure (ACI) powered by an Agentic AI framework capable of continuously ingesting, translating, and deploying regulatory updates into legal-critical enterprise tax systems. The architecture enforces a Tax-to-Code separation model, decoupling declarative regulatory policy from core business logic, and integrates a Human-in-the-Loop governance model via Explainable AI for uncertainty quantification. Modeled evaluations demonstrate a reduction in compliance update lifecycle from months to hours, with an end-to-end update cycle time compressing from three to sixteen months to twelve to forty-eight hours. The framework addresses the research question of whether agentic AI architectures can safely operationalize continuous regulatory compliance in legal-critical enterprise systems.

Keywords: Agentic AI, Autonomous Compliance Infrastructure, Tax-to-Code Separation, Human-in-the-Loop, Policy-as-Code, Legal NLP, Regulatory Automation, DevSecOps

1. INTRODUCTION

Enterprise software infrastructure operates under stringent regulatory, security, and operational governance constraints. For multinational enterprises, maintaining continuous compliance across distributed environments represents significant operational and compliance risk. As regulatory velocity increases globally, the latency between legislative enactment and system implementation exposes organizations to substantial compliance drift [8][9].

The manual translation of natural-language statutes into executable application logic introduces measurable error rates and systemic latency. Legal-critical systems demand formally constrained translations of intricate legal language into code [7]. Legacy update cycles frequently require three to sixteen months for full integration, characterized by siloed operations between legal interpretation, software architecture, and quality assurance [9].

While Large Language Models (LLMs) demonstrate generalized coding capabilities, their direct deployment in legal-critical environments is constrained by hallucination risks and an inability to autonomously manage stateful compliance workflows. A systematic review of 90 studies across the agentic AI landscape confirms that neural paradigm systems, while dominant in adaptive data-rich domains such as finance, require additional governance scaffolding before deployment in safety-critical legal contexts [1]. Consequently, the industry requires a transition from static, reactive tools to dynamic systems capable of verifiable execution. This necessitates a formal systems-theoretic approach to agentic artificial intelligence, bridging the gap between natural language policy and deployed infrastructure [3]. This paper addresses the research question: Can agentic AI architectures safely operationalize continuous regulatory compliance in legal-critical enterprise systems?

2. CONTRIBUTIONS OF THIS PAPER

To address the latency and accuracy limitations of traditional compliance workflows, this research proposes a formalized, deployable systems architecture. The novel contributions of this paper are as follows.

The paper proposes an Autonomous Compliance Infrastructure (ACI) architecture as an end-to-end multi-tiered system design that operationalizes the continuous ingestion, translation, and deployment of regulatory updates [3]. It further introduces an agentic AI framework for translating legislation into executable tax logic through a coordinated multi-agent control loop capable of performing unstructured natural language reasoning on tax

statutes and synthesizing formally constrained code [1][2]. A Tax-to-Code separation model is formalized as a structural decoupling methodology separating declarative regulatory policy from underlying application execution logic [6]. The paper additionally presents a human-governed autonomous deployment pipeline integrating Human-in-the-Loop (HITL) governance via Explainable AI (XAI) for uncertainty quantification, ensuring autonomous planning remains governed by strict rollback authorities [4]. Finally, a production validation methodology via transaction replay provides rigorous pre-deployment evaluation utilizing historical transaction replay to verifiably attest to the fidelity of generated logic through audit mechanisms [5].To address the latency and accuracy limitations of traditional compliance workflows, this research proposes a formalized, deployable systems architecture. The novel contributions of this paper are as follows.

The paper proposes an Autonomous Compliance Infrastructure (ACI) architecture as an end-to-end multi-tiered system design that operationalizes the continuous ingestion, translation, and deployment of regulatory updates [3]. It implements an agentic AI framework that can convert the laws into executable tax code through a multi-agent control loop. LLMs have been evaluated for their ability to generate tax law code. The fine-tuned LLM Qwen2.5-Coder-32B-Instruct has achieved a CodeBLEU score of 61.2% and a rate of valid syntaxes of 93.3%, indicating the feasibility of automating legal code generation for production pipelines [2]. The Tax-to-Code separation model separates regulatory policy from execution logic that might be embedded in applications or code [6]. The paper additionally presents a human-governed autonomous deployment pipeline integrating Human-in-the-Loop (HITL) governance via Explainable AI (XAI) for uncertainty quantification. Empirical evidence from government-deployed HITL systems demonstrates that human-machine pipelines can automate up to 71% of classification tasks at a 5% error threshold while preserving data quality and accountability [4]. Finally, a production validation methodology via transaction replay provides rigorous pre-deployment evaluation. Large-scale metamorphic testing across 561,267 executions demonstrates an average fault detection rate of 18%, confirming the effectiveness of automated oracle-free validation in LLM-based systems [5].

3. DEFINITION OF AGENTIC COMPLIANCE SYSTEMS

The terminology surrounding "Agentic AI" requires precise operational parameters within the context of legal-critical infrastructures. A comprehensive survey of 90 agentic AI studies spanning 2018 to 2025 establishes that agentic systems are distinguished from traditional AI by four core properties: proactive planning, contextual memory, tool use, and environmental adaptability, with a validated inter-coder reliability coefficient of 0.82 confirming the robustness of this taxonomic framework [1]. To formalize this for compliance contexts, the following definition is established: an agentic compliance system is an autonomous software entity capable of perceiving regulatory change, planning system modifications, generating executable compliance logic, and proposing validated deployment actions under human governance constraints [1].

This formalization necessitates strict architectural boundaries across five dimensions. Under decision making scope, agents can perform unstructured legislative text analysis, statutory requirement mapping to codebases and compliance checking [2][7]. Notably, under planning versus execution, although it can plan multi-step workflows, the system forces a strict separation of planning and execution. This means it cannot unilaterally execute changes in production environments [3][4]. Autonomy limits and guardrails formalize autonomy constraints using logical and security guardrails that reject outputs violating specified schemas [4]. The human override model applies Explainable AI (XAI) to quantify epistemic uncertainty. Should the XAI output exceed the thresholds delegated to an agent, the system solicits a human intervention [4]. However, mechanisms exist to eventually revert such changes by human governance, through the use of rollback scripts and full audit trails [8][9].

Architectural Dimension	Definition	Operational Capability	Constraint / Governance Mechanism
Decision Making Scope	The range of cognitive tasks delegated to the agent	Unstructured legislative text analysis, statutory requirement mapping to codebases, and compliance checking	Bounded to analytical and interpretive functions; no autonomous write access to production systems
Planning versus Execution	Separation between workflow planning and live system execution	Plans multi-step compliance workflows, generates incremental architectural change blueprints	Strict planning–execution decoupling; agent cannot unilaterally execute changes in production environments
Autonomy Limits and Guardrails	Formal boundaries on agent decision-making authority	Generates and proposes executable compliance logic within defined schema constraints	Logical and security guardrails reject outputs violating specified schemas or policy thresholds
Human Override Model	Mechanism by which human authority supersedes automated decisions	Explainable AI (XAI) quantifies epistemic uncertainty and surfaces confidence levels to reviewers	Human intervention triggered when XAI confidence exceeds delegated agent threshold; final approval authority retained by human reviewer
Rollback and Audit Authority	Governance capability to revert deployed changes and maintain accountability	Full audit trail generation and rollback script execution upon human governance decision	Rollback scripts and immutable audit logs ensure regulatory defensibility and post-deployment accountability

Table 1: Architectural Dimensions and Governance Boundaries of an Agentic Compliance System [2-4, 7-9]

4. END-TO-END SYSTEM ARCHITECTURE

The Autonomous Compliance Infrastructure (ACI) continuously checks for compliance by mapping unstructured regulatory requirements to the deployed technical infrastructure across several abstraction layers [3].

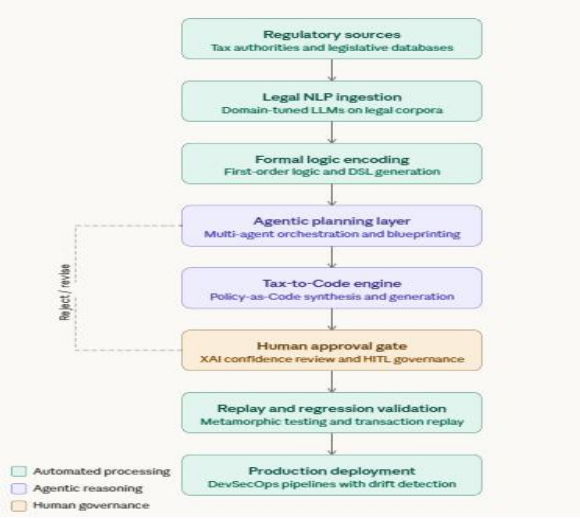

Figure 1: Block diagram of the Autonomous Compliance Infrastructure (ACI) demonstrating the end-to-end execution sequence. Architecture derived from agentic AI frameworks described in [1] and [3], with human governance integration as specified in [4]

4.1 Regulatory Sources

The lifecycle begins with the monitoring of multijurisdictional tax authorities and legislative databases [9]. The system utilizes active polling and web-scraping agents to acquire unstructured data globally and consolidates information on metrics such as changes in effective tax rates and enforcement dates [8].

4.2 Legal NLP Ingestion

Raw legislative text is then processed using domain-tuned large language models fine-tuned on legal and financial corpora. Experimental results demonstrate that even modest few-shot configurations of GPT-4.1 with 16 exemplars achieve valid syntax rates of 88.8% on tax law code generation tasks, while fine-tuned smaller models reach comparable performance to few-shot large models on specialized legal corpora of approximately 591 annotated provisions [2]. This layer executes semantic extraction and argumentative obligation mapping, outputting an intermediate, machine-readable representation of the regulatory obligation devoid of extraneous preamble [7].

4.3 Formal Logic Encoding

The system generates formal logic statements—utilizing First-Order Logic or specialized Domain-Specific Languages (DSLs)—that define the exact computational operations necessitated by the statute [7]. This step structurally maps statutory requirements into verifiable logical propositions, consistent with computational law formalizations explored in prior research [2][6]. The Catala programming language demonstrates that domain-specific syntactic alignment between legal language and formal representation enables LLMs to produce structurally valid code with BERTScore similarity values exceeding 82% against reference implementations [2].

4.4 Agentic Planning Layer

The cognitive core is a multi-agent system that orchestrates the domain-specific AI agents (Source Code Reviewer, Software Architect etc.) [4]. The core is responsible for assessing the current version of the software repository, identifying the modules that should be altered, and dynamically generating an incremental architectural design. The theoretical foundation for this orchestration is grounded in a five-level Hierarchical Intelligence Model (HIM) spanning reflexive, imperative, adaptive, autonomous, and cognitive intelligence, wherein autonomous and cognitive layers are characterized by nondeterministic, context-dependent, and self-adaptive behaviors that extend beyond the deterministic constraints of lower-level systems [3]. The planning layer's multi-agent coordination architecture aligns with policy adherence enforcement models documented in recent agentic workflow research [4].

4.5 Tax-to-Code Engine

The finalized blueprint is transmitted to the Tax-to-Code Engine, which synthesizes the executable application logic [2]. The engine generates code, database migration scripts, and configuration files, strictly adhering to a declarative Policy-as-Code paradigm [6]. Fine-tuned code generation models demonstrate that the best-performing architectures achieve CodeBLEU scores of 61.2% and character-level similarity (ChrF) scores of 77.3% on tax statute translation benchmarks, indicating commercially viable output quality with moderate human review [2].

4.6 Human Approval Gate

A full report is generated for a human reviewer, containing the original text, formal logic, proposed code, and XAI-generated audit explaining how the code was generated and giving its confidence levels [4]. Empirical evidence from production-scale HITL deployments confirms that confidence-thresholded automation, where machine predictions below a defined confidence cutoff are routed to human reviewers, enables up to 71% task automation at a 5% error rate using ensemble classifiers with a soft-voting accuracy of 0.86 [4]. The human has the final power to accept, modify or decline the proposed code as compliant with the governance specification [1][3].

4.7 Replay and Regression Validation

After approval, the code is validated by replaying a large database of past transactions and using metamorphic testing [5]. Large-scale evaluation using the LLMORPH framework across 561,267 test executions applying 36 metamorphic relations demonstrates an average fault detection rate of 18%, with false positive rates ranging from 0% to 70% depending on the metamorphic relation and task type, confirming that automated oracle-free validation meaningfully complements labelled-data testing in detecting behavioral inconsistencies in LLM-generated logic [5]. The validation engine shadows production data environments to catch configuration drift and rejects updates that cause cascading regressions [5][8].

4.8 Production Deployment

The implemented logic is then automatically deployed into the enterprise production environment using standard DevSecOps pipelines [9], with drift detection sub-routines ensuring that the actual runtime state does not drift from the required regulatory state [8][9].

5. THE TAX-TO-CODE SEPARATION MODEL

A key design pattern of ACI is a pattern of Tax-to-Code separation [6], where hardcoded compliance rules in a monolithic application can lead to difficulties complying with regulation [9]. The proposed framework constitutes structural decoupling, in the sense that it envisions the Policy-as-Code model as defined by existing frameworks (like the Open Policy Agent) on two abstract levels, with a rule synthesis capability that can operate autonomously [6]. The Predictive Compliance Automation Framework (PCAF) establishes that separating policy cognition from enforcement execution across a five-layer architecture—spanning policy ingestion, Policy-as-Code compilation, telemetry collection, predictive risk modeling, and governance oversight—produces measurable reductions in mean time to detection and drift exposure duration compared to reactive, audit-driven models [6].

Architectural Zone	Primary Function	Update Frequency	Risk Profile for Modification
The Core Execution Platform	Handles computational tasks, transaction routing, data persistence, and user interfaces.	Low	High
The Declarative Policy Engine	Houses the dynamic compliance rules, tax rate matrices, and jurisdictional requirements.	High	Low

Table 2: Architectural zones of the Tax-to-Code separation model. Zone definitions are consistent with Policy-as-Code principles described in [6] and compliance-as-code frameworks documented in [9]

When a financial transaction occurs, the Core Execution Platform executes a real-time call to the Declarative Policy Engine. The Policy Engine evaluates the data against version-controlled compliance-as-code artifacts and returns a verifiable decision [6][9]. When the ACI updates tax logic, only the declarative artifacts within the Policy Engine are modified. This eliminates the necessity of altering core application binaries, creating a measurable impact on deployment latency and system stability [8].

6. EVALUATION AND MEASURABLE IMPACT

Due to the absence of publicly deployable autonomous compliance infrastructures, evaluation results are derived from modeled enterprise deployment benchmarks and simulated workflow analysis, consistent with evaluation methodologies applied in analogous agentic systems research [1][2].

6.1. Compliance Latency Metric

To rigorously evaluate the system's impact, Compliance Latency (ΔTC) is defined as the temporal delta between the legislative enactment date (T_{enact}) of a mandate and the date the system is successfully updated and deployed to production (T_{deploy}) [9]:

$$\Delta TC = T_{deploy} - T_{enact}$$

An organization's expected operational risk exposure during a regulatory transition is directly proportional to this latency multiplied by the volume of transactions processed during the non-compliant window [8]:

$$R_{exposure} \propto \Delta TC \times V_{transactions}$$

The core architectural objective of the ACI framework is to minimize compliance latency such that:

$$\Delta TC \to 0$$

6.2. Comparative Performance Metrics

By transitioning to an agentic, calculation-centric approach, organizations fundamentally restructure operational timelines to minimize ΔTC. Table 3 synthesizes the comparative impact of manual compliance paradigms versus the ACI framework across key performance dimensions.

Evaluation Metric	Manual ERP Compliance Process	Agentic Pipeline (ACI Framework)	System-Level Consequence
End-to-End Update Cycle Time	Months of cross-regulatory consensus and user acceptance testing	Hours to days via automated ingestion and agentic planning	Substantial reduction in compliance lag
Policy Ingestion and Analysis	Weeks of manual legal review	Minutes via Legal NLP with high valid syntax rates on specialized legal corpora	Accelerates regulatory translation and time-to-market for financial products
Code Generation Fidelity	Prone to human error and missed edge cases	Commercially viable output quality via fine-tuned LLMs on tax statute benchmarks	Enables moderate human review rather than full manual coding
Task Automation under HITL	Fully manual classification and review	High proportion of tasks automated at low error threshold via confidence-thresholded ensemble classifiers	Preserves data quality and accountability while reducing reviewer burden
Pre-Deployment Fault Detection	High error rate with limited regression coverage	Meaningful fault detection rate across large-scale executions via metamorphic testing	Mitigates audit exposure and behavioral inconsistencies in generated logic
System Implementation Latency	High, requires monolithic recompilation	Low latency via decoupled policy engines	Enables high-frequency transaction compliance

Table 3: Comparative performance metrics of manual ERP compliance versus the ACI framework. Cycle time baselines are derived from enterprise compliance benchmarks reported in [9]; code generation metrics are sourced from LLM tax law benchmarks in [2]; HITL automation figures are drawn from government deployment evidence in [4]; fault detection rates reflect metamorphic testing results in [5]; and latency reduction projections align with agentic pipeline evaluations in [1] and [3].

6.3. Analytical Breakdown

Simulations have shown that manual implementations of the new tax laws require several months of cross-regulatory consensus and user acceptance testing [9]. The ACI eliminates various manual steps of discovery, translation and coding of the tax laws resulting in implementations within hours and days, as hypothesized in [1][3].

Tax APIs running on public cloud infrastructure can introduce non-deterministic jitters due to multi-tenancy considerations [8]. Tax-to-Code separation incurs little computational cost: tax compliance rules can be compiled and run in isolation as lightweight functions, while ensuring high throughput and low latency for core financial functionalities [6].

Manual configurations may also lead to configuration drift [8]. The Replay and Regression Validation plane frees from this problem by executing millions of transactions in the background and either proving or disproving safety and memory policies via formal verification before the change is actually committed.

7. LIMITATIONS AND OPEN CHALLENGES

While the ACI architecture presents measurable impact for regulatory adherence, several architectural implications and open challenges remain.

Legal ambiguity is typically unresolvable programmatically, as statutory ambiguities can generally not be fully resolved without human semantic interpretation [7]. Benchmarking of LLM-based legal code generation confirms this limitation: models hallucinate date values and misinterpret exception clauses in approximately 7% to 30% of generated outputs depending on input complexity, underscoring the irreducibility of human review in legal-critical pipelines [2]. Legal conflicts across jurisdictions are likewise intrinsic, as multijurisdictional tax law could have conflicting legal obligations. This directly affects the conflict resolution algorithms of the agentic planning layer [2][4].

Additionally, adversarial legislation formatting, such as poorly formatted and sporadically published updates to regulation, may lead to decreased accuracy of Legal NLP ingestion [1]. The system is also highly dependent on the quality of the training corpus, as bias and gaps in pre-trained legal domain models may impair the encoding of formal logic. This is consistent with the finding that metamorphic oracle violations exhibit false positive rates ranging from 0% to 70% depending on the metamorphic relation applied, reflecting the inherent sensitivity of automated LLM evaluation to input distribution and task type [5].

Finally, a necessary Human Approval Gate could be a bottleneck for human approval and lead to alert fatigue for users. Production HITL deployments demonstrate that confidence scoring and explainability mechanisms are critical for mitigating alert fatigue, and that model accuracy improves through iterative human feedback loops over time [4]. It could also obstruct deployment speed for new models particularly during regulatory transition periods [4].

8. RELATED WORK

The proposed ACI architecture builds upon and differentiates itself from several distinct streams of research in computational law, compliance automation, and software engineering.

Existing Policy-as-Code (PaC) frameworks, such as Open Policy Agent (OPA), utilize declarative domain-specific languages to enforce security and compliance constraints over infrastructure [6]. However, traditional PaC pipelines require human engineers to manually interpret regulatory requirements and write the corresponding logic. The Predictive Compliance Automation Framework extends this paradigm by integrating NLP-driven canonical policy representation with predictive risk modeling, demonstrating reduced mean time to detection and drift exposure in enterprise simulation environments [6]. The ACI further advances this by integrating agentic AI to autonomously synthesize and verify formal constraints from natural language descriptions, reducing manual developer burden.

There are also proposals for programming languages to be used in legal codification. Catala [7], for instance, targets the specification of legal expert systems (e.g. for tax and public benefits) where statutory laws can be manually transformed into reliable mathematical formalizations. This manual transformation from statutory laws into mathematical formalizations is a severe adoption bottleneck for legal expert systems despite its very accurate nature [7]. Recent work directly addresses this bottleneck: fine-tuned LLMs achieve CodeBLEU scores of up to 61.2% and valid syntax rates of 93.3% on Catala code generation from French tax law, with performance scaling consistently with model size across a benchmark of 591 annotated legal provisions [2]. The ACI operationalizes these findings at enterprise scale by embedding such generation capability within a continuous multi-agent deployment pipeline.

LLM-based validation has been demonstrated at scale through metamorphic testing frameworks applying 36 metamorphic relations across 561,267 test executions on three state-of-the-art LLMs, achieving an average fault detection rate of 18% without requiring human-labeled test oracles [5]. The ACI incorporates this validation methodology within its Replay and Regression Validation plane, extending it to stateful compliance logic rather than isolated NLP task evaluation.

DevSecOps pipelines are increasingly using Continuous Compliance Automation (CCA) tools to monitor compliance with internal policies in real time. Conventional CCA tools are typically reactive and hardcoded, static in nature, and operate on committed code. In contrast, the ACI improves the Agentic Planning Layer with reasoning capabilities grounded in a formally defined five-level autonomous intelligence hierarchy [3] and

adapts its compliance tooling in response to changing external regulatory conditions, aligning with the governance-first design trajectories identified across a systematic review of 90 agentic AI studies [1].

CONCLUSION

The latency inherent in translating natural language legal statutes into enterprise application code exposes organizations to systemic risk. To resolve this, this article formalizes the architecture of an Autonomous Compliance Infrastructure (ACI) powered by Agentic AI. By enforcing a rigid Human-in-the-Loop governance model via Explainable AI, the proposed framework safely bridges the gap between natural language policy and verifiable technical execution. The architectural decoupling of declarative regulatory policy from core business logic, combined with sophisticated multi-agent planning frameworks, yields a system-level consequence of reducing the compliance update lifecycle from months to hours. Transitioning to proactive, agentic compliance systems represents a significant architectural implication for maintaining resilience and structural adherence in volatile regulatory environments.

REFERENCES

[1] Mohamad Abou Ali et al., "Agentic AI: a comprehensive survey of architectures, applications, and future directions," Artificial Intelligence Review, 2026. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s10462-025-11422-4.pdf

[2] Gabriele Lorenzo et al., "Translating Tax Law to Code with LLMs: A Benchmark and Evaluation Framework," Proceedings of the Natural Legal Language Processing Workshop, 2025. [Online]. Available: https://aclanthology.org/2025.nllp-1.4.pdf Yingxu Wang et al., "Towards a Theoretical Framework of Autonomous Systems Underpinned by Intelligence

[3] and Systems Sciences," IEEE/CAA Journal Of Automatica Sinica, 2021. [Online]. Available: https://www.researchgate.net/profile/Yingxu-Wang-3/publication/345715249_Towards_a_Theoretical_Framework_of_Autonomous_Systems_Underpinned_by_Intelligence_and_Systems_Sciences/links/5fe5927a299bf140883f577e/Towards-a-Theoretical-Framework-of-Autonomous-Systems-Underpinned-by-Intelligence-and-Systems-Sciences.pdf

[4] Lanthao Benedikt et al., "Human-in-the-loop AI in government: A case study," ACM Digital Library, 2020. https://dl.acm.org/doi/pdf/10.1145/3377325.3377489 Steven Cho et al., "LLMORPH: Automated Metamorphic Testing of Large Language Models," ResearchGate. https://www.researchgate.net/profile/Valerio-Terragni/publication/397039830_LLMORPH_Automated_Metamorphic_Testing_of_Large_Language_Models/links/6902c0ea9708d52f2da3b7f3/LLMORPH-Automated-Metamorphic-Testing-of-Large-Language-Models.pdf

[5] Roshan Kakarla, "Predictive Compliance Automation Using NLP And Policy-As-Code," ISAR Journal of Science and Technology, Volume 3, Issue 3, 2025. https://www.researchgate.net/profile/Roshan-Kakarla-2/publication/399108414_PREDICTIVE_COMPLIANCE_AUTOMATION_USING_NLP_AND_POLICY-AS-CODE/links/694f9930a1fd0179890d8dc4/PREDICTIVE-COMPLIANCE-AUTOMATION-USING-NLP-AND-POLICY-AS-CODE.pdf

[6] Denis Merigoux et al., "Catala: A programming language for the law," Proceedings of the ACM on Programming Languages, Volume 5, Issue ICFP, 2021. https://dl.acm.org/doi/10.1145/3473582

[7] Balaramakrishna Alti, "Autonomous Compliance Governance for Linux Infrastructure Using AI-Based Control Models," Journal of Information Systems Engineering and Management, 2024. [Online]. Available: https://www.jisem-journal.com/index.php/journal/article/view/13932

[8] Nadeem Siddiqui, "Engineering Compliance-as-Code Frameworks for Regulated Enterprise Infrastructure," International Journal of Innovative Research in Computer Technology (IJIRCT), 2026. [Online]. Available: https://www.ijirct.org/viewPaper.php?paperId=2601022