Article

Not peer-reviewed version

# Operationally Audit-Ready Dual-Flow Compliance Pipelines for Conformance Matrices: An Ontology-Based Metamodel with GDPR and EU AI Act Instantiation

Antonio Goncalves * and Anacleto Correia

Posted Date: 26 January 2026

doi: 10.20944/preprints202601.1812.v1

Keywords: AI governance; Compliance-by-Design (CbD); operational audit readiness; compliance pipeline; conformance matrices; ontology-based metamodel; evidence traceability; General Data Protection Regulation (GDPR); European Union Artificial Intelligence Act (EU AI Act); Data Privacy Vocabulary (DPV); AI Risk Ontology (AIRO)}

*Article*

# Operationally Audit-Ready Dual-Flow Compliance Pipelines for Conformance Matrices: An Ontology-Based Metamodel with GDPR and EU AI Act Instantiation

**Antonio Goncalves * and Anacleto Correia**

Centro de Investigação Naval (CINAV), 2810-001 Almada, Portugal
* Correspondence: agoncalveslx@gmail.com

**Abstract**

Artificial intelligence (AI) risk systems deployed in high-stakes decision-support settings are increasingly expected to be *operationally audit-ready*: they must demonstrate, through verifiable evidence, that applicable governance requirements were implemented, monitored, and maintained during real-world operation. However, audit readiness frequently breaks down not due to missing documentation, but because the trace links between normative requirements, operational controls, and both pipeline artefacts and evidence items are fragmented, inconsistent, and costly to verify. To address this gap, this paper establishes a foundation for *audit-ready conformance matrices* grounded on a **dual-flow, layered architecture** that combines an upstream technical pipeline with a downstream **compliance pipeline** designed to operationalise requirements as explicit controls, evidence specifications, gates, corrective actions, and accountability hooks. The approach delivers five core artefacts: (i) an ontology-aligned interoperability layer leveraging the **Data Privacy Vocabulary (DPV)** and the **AI Risk Ontology (AIRO)**; (ii) a conformance-matrix **metamodel** defining the entities and relations needed to represent requirements, controls, artefacts, and evidence; (iii) mapping rules that bind controls to concrete operational artefacts and evidence items; (iv) a case-by-case **instantiation workflow** producing distinct matrix instances for distinct pipelines and contexts; and (v) a multi-regime alignment mechanism. While *multi-regime by design*, the paper provides a primary instantiation for the **General Data Protection Regulation (GDPR)** and the **European Union Artificial Intelligence Act (EU AI Act)**. Conceptual validation is provided through competency questions, consistency checks, and an illustrative instantiation over an AI risk pipeline. Overall, the work reframes **Compliance-by-Design (CbD)** as an operational property supported by reusable, auditable trace structures rather than retroactive reporting.

**Keywords:** AI governance; Compliance-by-Design (CbD); operational audit readiness; compliance pipeline; conformance matrices; ontology-based metamodel; evidence traceability; General Data Protection Regulation (GDPR); European Union Artificial Intelligence Act (EU AI Act); Data Privacy Vocabulary (DPV); AI Risk Ontology (AIRO)

## 1. Introduction

High-stakes decision-support systems that incorporate artificial intelligence (AI) are increasingly assessed not only on predictive performance but also on whether they can remain *operationally audit-ready* across their lifecycle, i.e., able to demonstrate —with verifiable, reviewable evidence—that applicable governance requirements were implemented, monitored, and maintained during real-world operation. In the European Union, this expectation is reinforced by binding requirements for lawful and accountable processing of personal data and, increasingly, by system-level obligations for high-risk AI. These pressures are reflected in the General Data Protection Regulation (GDPR) and in the European Union Artificial Intelligence Act (EU AI Act), as well as in risk- and control-orientated

 doi:10.20944/preprints202601.1812.v1



governance frameworks such as the NIST Artificial Intelligence Risk Management Framework (AI RMF). [1–3] Yet, experience shows that accountability can fail even when organisations can point to documentation, because the evidence needed to support claims about controls and their continuous application is incomplete, inconsistent, or not bound to the concrete system artefacts available for scrutiny. [4]

A core reason is that conventional technical pipelines—covering data preparation, training, evaluation, packaging, and deployment—are not designed to provide end-to-end traceability between *normative requirements* (e.g., legal duties, internal policies, or standard controls) and the concrete artefacts produced and used by the system. Software engineering for machine learning practices do introduce lifecycle disciplines (e.g., staged releases, quality gates, and monitoring), but they primarily optimise delivery, reliability, and maintainability; they do not systematically encode governance requirements as operational controls with evidence bindings. [5,6] As a result, the trace links needed for audit and assurance remain fragmented across tools, teams, and artefact types, and are expensive to reconstruct post hoc. [7]

Operational audit readiness, therefore, requires an explicit, evidence-centric layer that treats governance requirements as *operational constraints* and transforms them into verifiable controls with measurable checkpoints, defined evidence items, and corrective actions. A practical prerequisite is that evidence items are unambiguously identifiable and comparable across runs and releases, which motivates the use of shared provenance and interoperability primitives. In particular, the W3C PROV model provides a widely used vocabulary to represent provenance relations between entities, activities, and agents, supporting reproducible and reviewable evidence trails across heterogeneous artefacts. [8]

To address this gap, we adopt a *dual-flow*, layered architecture that separates concerns while maintaining explicit trace links. The first flow is a conventional upstream technical pipeline (data $\rightarrow$ training $\rightarrow$ evaluation $\rightarrow$ deployment). The second flow —the contribution of this paper—is a downstream **compliance pipeline** engineered to operationalise governance requirements as follows: (i) explicit controls; (ii) evidence specifications; (iii) gates that condition transitions between lifecycle stages; and (iv) corrective actions that close the loop when checks fail. The core operational output is an *audit-ready conformance matrix* that connects requirements to controls and binds each control to concrete pipeline artefacts and evidence items for a given context and release. A prerequisite for such matrices is terminological consistency across stakeholders, artefacts, and regimes; we therefore ground the approach in an ontology-aligned vocabulary, leveraging the Data Privacy Vocabulary (DPV) and the AI Risk Ontology (AIRO) to support interoperable descriptions of processing, risk, and controls across matrix instances. [9,10]

To make the motivation explicit and operational, Table 1 summarises recurring audit-readiness failure modes and the corresponding architectural responses introduced in this work.

**Table 1.** Problem–gap–consequence–response summary for operational audit readiness.

| Observed problem | Why the technical pipeline is insufficient | Audit/assurance consequence | Response in this paper |
|---|---|---|---|
| Evidence is dispersed across tools and teams | Normative requirements are not encoded as operational controls with stable trace links | High reconstruction cost; low verifiability | Dual-flow architecture + conformance-matrix meta-model linking requirements–controls–artefacts–evidence |
| Controls are documented but not operationalised | No explicit gates, decision records, or acceptance criteria coupled to lifecycle events | Retroactive compliance; weak assurance claims | Compliance flow with gates, decision records, and a corrective-action loop (assurance layer) |
| Terminology varies across stakeholders and artefacts | Semantic heterogeneity makes mappings brittle and inconsistent | Inconsistent matrices; poor interoperability | Ontology-aligned vocabulary (DPV, AIRO) to stabilise concepts across instantiations |
| Continuous change (data/model/deployment) invalidates prior evidence | Evidence is not run-scoped, versioned, and integrity-anchored | Stale evidence; audit findings hard to reproduce | Run-level evidence bundles, stable identifiers, integrity metadata, and traceability checks |

 doi:10.20944/preprints202601.1812.v1



This framing clarifies why conventional technical pipelines are necessary but insufficient for operational audit readiness.

### 1.1. Contributions

This paper provides a reusable foundation for operational audit readiness by design, centred on conformance matrices as first-class, auditable artefacts:

- **C1 — Canonical dual-flow framing**: We formalise a dual-flow architecture that distinguishes the upstream technical pipeline from a downstream compliance pipeline engineered for operational audit readiness.
- **C2 — Ontology-grounded vocabulary**: We specify an ontology-aligned interoperability layer, leveraging DPV and AIRO, to support consistent terminology across conformance matrices and evidence records.
- **C3 — Compliance-matrix metamodel**: We define a metamodel capturing requirements, controls, artefacts, and evidence items with explicit trace links, stable identifiers, and validation constraints.
- **C4 — Case-by-case instantiation workflow**: We provide an instantiation workflow that produces context-specific conformance matrix instances and supports multi-regime alignment, with a primary instantiation for GDPR and the EU AI Act.
- **C5 — Verification and quality checks**: We propose competency questions and consistency checks that assess the completeness and traceability of a given matrix instance and its evidence bindings.

### 1.2. Paper Organisation

The remainder of this paper is organised as follows. Section 2 introduces the canonical dual-flow architecture and problem framing. Section 3 situates the work in the relevant background and related work. Section 4 outlines the normative scope and requirement types considered. Section 5 presents the ontology vocabulary, conformance-matrix metamodel, and instantiation workflow. Section 6 specifies the evidence model and audit queries. Sections 7–9 report implementation, evaluation, and discussion, and Section 10 concludes the paper.

## 2. Canonical Dual-Flow Architecture and Problem Framing

### 2.1. Dual-Flow, Layered Architecture

We adopt a **layered, dual-flow architecture** composed of two tightly coupled pipelines: (i) a **conventional** technical machine learning (ML) pipeline and (ii) a **compliance pipeline** engineered to operationalise governance requirements throughout the system lifecycle. As illustrated in Figure 1, the technical flow progresses through five layers—Data, **Feature**, **Model**, **Deployment**, and **Monitor—that** reflect established production practices and widely reported engineering challenges in ML systems, including iterative change, dependency coupling, and the accumulation of technical debt [5,6]. In this manuscript, the technical pipeline is treated as a **baseline execution context**; it is not presented as novel.

The **novel contribution** of this work lies in the second flow: a **compliance pipeline** designed as an **operational** layer rather than as an afterthought for documentation. In Figure 1, the compliance flow is structured into five layers with distinct responsibilities:

- **Scope**: Define the system boundary, intended use, relevant data flows, and the set of applicable regimes and organisational policies (primary instantiation: GDPR and EU AI Act).
- **Requirements**: derive and record the applicable normative requirements under the defined scope, including applicability conditions and scoping assumptions.
- **Controls**: translate requirements into verifiable control objectives and implementable controls with stable identifiers, ownership, and verification intent.
- **Evidence**: specify the minimal evidence required for each control (required artefact types, checks, and acceptance criteria) and bind evidence expectations to lifecycle touchpoints in the technical flow.
- **Assurance**: Operationalise verification through gates and decision records, producing pass/-fail/incomplete statuses per control and conditioning lifecycle progression on evidence sufficiency.

 doi:10.20944/preprints202601.1812.v1



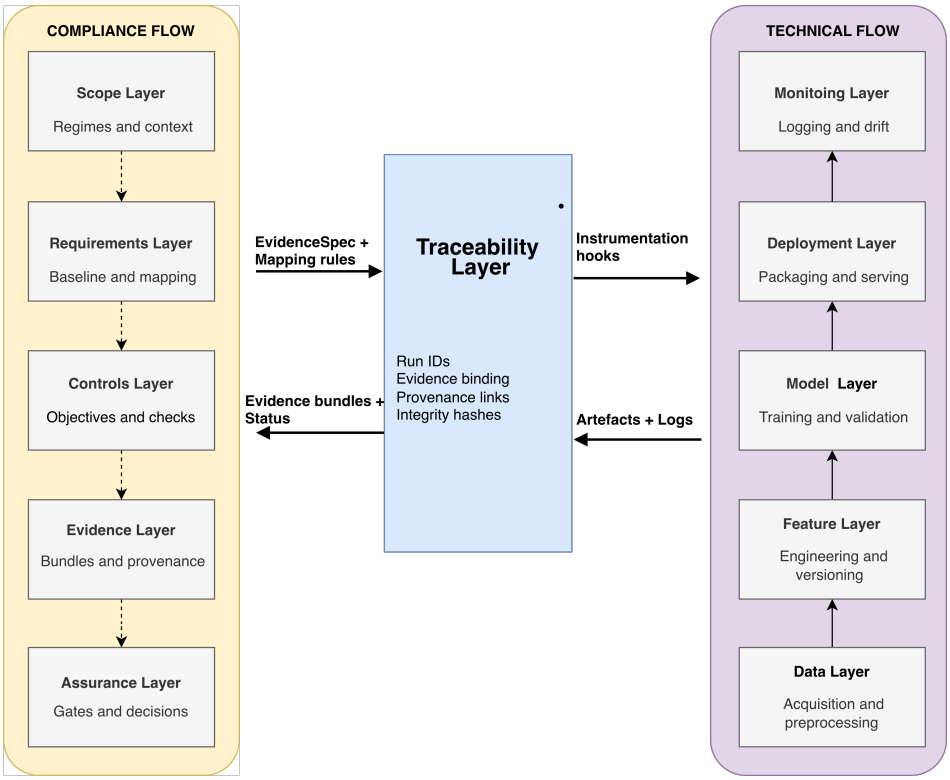


**Figure 1.** Dual-flow, layered architecture: a vertical technical flow (Data–Feature–Model–Deployment–Monitor) coupled with a vertical compliance flow (Scope–Requirements–Controls–Evidence–Assurance) through a Traceability Layer that binds run identifiers, pipeline artefacts, and evidence items. The operational outputs are audit-ready conformance matrices and run-level evidence bundles.

Crucially, the two flows are coupled through the central **Traceability Layer** (Figure 1), which acts as the operational interface between what the technical pipeline produces and what the compliance pipeline must verify. The Traceability Layer: (i) assigns and propagates **run identifiers** across the lifecycle; (ii) binds **Controls** and **Evidence** specifications to concrete **pipeline artefacts** across the Data, Feature, Model, Deployment, and Monitor layers; and (iii) enables deterministic retrieval of **run-level evidence bundles** and their linkage to audit-ready conformance matrices. In this framing, compliance is treated as a set of **testable obligations coupled to lifecycle events** (e.g., dataset release, model promotion, deployment change), rather than as a post-hoc report.

To ensure enforceability, the **Assurance** layer implements gating mechanisms that condition the progression of the technical flow (e.g., promotion to production, roll-out, retraining approval) on meeting minimum evidence and acceptance criteria. Where explainability is used, it is treated only as an **optional evidence artefact** associated with specific controls, not as the centre of the architecture.

## 3. Background and Related Work

This paper sits at the intersection of software engineering for machine learning and operational AI governance. The relevant literature broadly agrees on two points. First, high-stakes machine learning systems require disciplined lifecycle practices because data, models, and configurations evolve continuously. Second, governance regimes and assurance programmes increasingly expect organisations to justify compliance claims with structured, checkable evidence. The persistent gap is that, even when documentation exists, the trace links between normative requirements, operational controls, pipeline artefacts, and evidence items remain fragmented and difficult to verify in a timely, repeatable way.

 doi:10.20944/preprints202601.1812.v1



### 3.1. Governance-by-Design and Assurance Across ML Lifecycles

Software engineering for machine learning emphasises that many real-world failures stem from integration and process issues rather than isolated model defects, and that technical debt accumulates when pipelines, data dependencies, and decision points are not explicitly managed [5,6]. In parallel, governance and accountability work stresses that auditability is an operational capability: it depends on how systems are built, deployed, monitored, and updated, and on whether the organisation can produce evidence that controls were applied as intended [4]. Practical governance guidance similarly frames assurance in terms of measurable controls and documentation/evidence outputs; for example, the National Institute of Standards and Technology (NIST) AI Risk Management Framework proposes governance functions that must be supported by actionable artefacts and procedures [3]. These streams motivate a design stance in which governance requirements are treated as engineering constraints that must be realised through pipeline mechanisms, not only written policies.

### 3.2. Traceability, Provenance, and Conformance Matrices

Requirements traceability has long been studied as a means to control change, support verification, and make accountability explicit in complex systems [11]. In compliance and assurance settings, trace structures are often implemented as matrices that map requirements to controls and to supporting artefacts. However, matrices assembled *post hoc* from heterogeneous materials (tickets, logs, reports, datasets, model versions) are costly to validate because the underlying evidence is dispersed, inconsistently described, and weakly linked to the control logic that produced it. Provenance standards address part of this problem by representing how entities, activities, and agents relate over time; the W3C PROV model provides a well-established basis for expressing such causal and responsibility relationships [8]. What remains under-specified in most operational settings is how to systematically connect (i) normative requirements, (ii) concrete controls, and (iii) provenance-anchored evidence into a reusable conformance structure that can be instantiated per pipeline and assessed by auditors.

### 3.3. Ontologies for Privacy, Risk, and Compliance Interoperability

A practical obstacle in operational governance is semantic heterogeneity: different teams and tools use inconsistent terminology for the same concepts (e.g., processing operations, risk categories, measures, evidence). Ontologies and controlled vocabularies address this by providing shared conceptual anchors for compliance artefacts and governance workflows. The *Data Privacy Vocabulary* supports consistent representation of privacy and data protection concepts that are frequently required in governance artefacts and registers [9]. For AI risk, the *AI Risk Ontology* provides a structured conceptualisation of risk-related entities that can be reused across risk assessment and assurance processes [12]. Recent work further highlights the practical audit gap between expanding normative obligations and the operational ability to evidence compliance in AI systems [13], and ethical assurance research argues for evidence-centric approaches designed for verification rather than narrative reporting [14]. Together, these works motivate an interoperability layer where requirements, controls, artefacts, and evidence can be expressed with shared semantics and queried consistently.

**Positioning and gap.** Building on these foundations, the specific contribution of this paper is to formalise the missing operational layer: a dual-flow architecture with an explicit compliance pipeline and an ontology-based conformance-matrix metamodel that binds requirements to controls, pipeline artefacts, and evidence items. This shifts compliance from retroactive documentation to a repeatable, checkable operational property while remaining compatible with established lifecycle engineering practices and ontology-driven interoperability.

## 4. Normative Scope and Requirements

This section sets the *normative scope* of the paper and derives a compact set of requirements that the proposed dual-flow architecture, ontology layer, conformance-matrix metamodel, and evidence model must satisfy. Our focus is **operational audit readiness**: the ability to produce verifiable, queryable, and

 doi:10.20944/preprints202601.1812.v1



consistently traceable links from normative obligations to implemented controls and their supporting evidence, without relying on costly post hoc reconstruction [4,13].

### 4.1. Scope and Assumptions

**Multi-regime by design, primary instantiation.** The framework is designed to support multiple regimes through explicit mappings and instantiation rules. In this paper, we provide a primary instantiation for the **General Data Protection Regulation (GDPR)** and the **European Union Artificial Intelligence Act (EU AI Act)**. The resulting matrices are *engineering artefacts for assurance and auditability*, not a claim of full legal compliance.

**Why the technical pipeline is insufficient.** Conventional technical pipelines (e.g., data preparation, training, evaluation, deployment) generate rich artefacts, but typically do not enforce *norm-to-artefact* traceability with stable identifiers, evidence specifications, and audit queries as first-class outputs. This gap is well documented in practice as a combination of accountability breakdowns and engineering drift in machine learning systems [5,6].

**Evidence and provenance premise.** We assume that auditability requires both (i) *pipeline artefacts* (e.g., model cards, dataset versions, evaluation reports, risk assessments) and (ii) *evidence items* that attest to control execution (e.g., approvals, gate decisions, logs, checksums). Provenance representations should be interoperable and machine-actionable, aligning with established provenance models [8]. Further, we assume auditors and assessors may have constrained visibility into system internals, which strengthens the need for structured, portable evidence bundles [15].

### 4.2. Derived Requirements

We derive requirements at two levels: (i) **functional requirements** that define what the compliance pipeline and conformance matrices must *do*; and (ii) **non-functional requirements** that constrain how these artefacts must behave under change, scale, and assessment pressure. Together, these requirements operationalise assurance-orientated governance patterns [3,14].

#### 4.2.1. Functional Requirements

**FR1 Norm-to-control formalisation.** Represent normative obligations as explicit control objectives with unambiguous identifiers, scope conditions, and accountable roles [4].

**FR2 Control-to-artefact mapping.** Bind each control objective to concrete pipeline artefacts and evidence items (what must exist, where it is produced, and how it is verified), enabling deterministic trace paths from requirements to artefacts.

**FR3 Evidence specification and gating.** For each control, define evidence specifications and gate conditions (pass/fail criteria, decision records, and corrective actions) so that conformance is assessed continuously rather than reconstructed retroactively [14].

**FR4 Matrix instantiation per context.** Support case-by-case instantiation that yields different conformance matrix instances for different pipelines, risk contexts, and applicable regimes (multi-regime by design).

**FR5 Queryable audit layer.** Enable auditors and assurance stakeholders to query trace links and evidence bundles using stable identifiers (e.g., by requirement, control, run, or lifecycle stage), including support for constrained-access assessment settings [15].

**FR6 Nonconformity handling.** Represent nonconformities, exceptions, and remediation actions as first-class entities linked to impacted controls and artefacts, supporting an auditable corrective loop [5].

#### 4.2.2. Non-Functional Requirements

**NFR1 Interoperability.** The vocabulary, mappings, and evidence descriptions should be exportable in interoperable forms, including provenance-aligned representations [8].

**NFR2 Extensibility.** The metamodel and mapping rules should admit new regim es and organisational controls without redesigning the core structure (multi-regime by design).

 doi:10.20944/preprints202601.1812.v1



**NFR3 Maintainability under drift.** The approach should remain usable under frequent model and data updates, limiting compliance "engineering debt" in evolving systems [6,16].

**NFR4 Auditability and integrity.** Evidence bundles should be tamper-evident and internally consistent (e.g., checksums, versioning, signed decisions), supporting third-party assessment and reproducibility of trace claims [8,13].

**NFR5 Reproducibility.** Re-running a given pipeline stage should reproduce the declared trace structure (even when outcomes change), reflecting established expectations in software engineering for machine learning [5].

The remainder of the paper instantiates these requirements as: an ontology-aligned vocabulary layer (§5), a conformance-matrix metamodel and mapping rules (§5), and an evidence model with audit queries (§6).

Table 2 provides a small, illustrative GDPR/EU AI Act slice to show how requirements are operationalised into control objectives, controls, evidence specifications, and gate-verifiable decisions; it is not intended as exhaustive legal coverage.

**Table 2.** Illustrative GDPR/EU AI Act instantiation slice (non-exhaustive): from requirement to gate-verifiable evidence.

| Normative anchor (high-level) | Control objective (CO) | Control (C) | Evidence specification (ES) | Gate |
|---|---|---|---|---|
| GDPR (lawful processing / accountability) | Demonstrate lawful basis and processing scope for in-scope data | Maintain processing register + approval record | Processing register entry; approval decision record; versioned scope assumptions | Data approval |
| GDPR (data minimisation / purpose limitation) | Ensure data fields and transformations are justified and documented | Dataset schema control + preprocessing lineage capture | Dataset schema; preprocessing lineage artefact; hash/timestamp; reviewer sign-off | Data release |
| EU AI Act (risk management for high-risk AI) | Maintain a risk register and mitigation trace for the system context | Risk assessment control + mitigation mapping | Risk register entry; mitigation-to-control mapping; review outcome | Model promotion |
| EU AI Act (monitoring / post-market duties) | Detect relevant drift/incidents and trigger corrective actions | Monitoring control + incident workflow control | Monitoring report; incident ticket; corrective-action record; re-verification result | Deployment/ rollback |

The same instantiation pattern generalises to additional regimes by extending mappings while preserving the core trace structure.

## 5. Ontology Vocabulary, Matrix Metamodel, and Instantiation

### *5.1. Ontology-Aligned Vocabulary for Interoperability*

Operational audit readiness requires that normative requirements can be traced to concrete controls and, ultimately, to verifiable evidence produced during system operation. In practice, traceability breaks when different stakeholders use inconsistent terminology for the same concepts (e.g., "risk" versus "harm", "processing" versus "use"), or when the same term is used with different meanings across regimes and organisational policies. To prevent ad hoc semantics and brittle mappings, we introduce an ontology-aligned vocabulary layer that provides stable, reusable semantics for (i) data-processing and privacy concepts and (ii) AI risk concepts. This layer does not replace legal texts; it provides a common semantic substrate that supports consistent control descriptions, evidence specifications, and audit queries.

 doi:10.20944/preprints202601.1812.v1



Minimal Profile and Reuse-First Strategy

We adopt a reuse-first strategy and define a *minimal profile* that includes only the concepts needed to (a) express control objectives, (b) name pipeline artefacts and evidence items, and (c) support audit queries over trace links. This minimises modelling overhead while preserving extensibility: additional regimes or domain-specific concepts can be introduced as optional modules without breaking existing trace structures.

DPV and AIRO as Complementary Anchors

For privacy and data-processing semantics, we leverage the **Data Privacy Vocabulary (DPV)** to represent actors, purposes, processing operations, data categories, legal bases, and safeguards [9,17]. For AI risk semantics, we leverage the **AI Risk Ontology (AIRO)** to represent risks, risk sources, impacts, affected stakeholders, and mitigations [10,12]. DPV and AIRO are complementary: DPV anchors obligations around personal data processing and governance measures, while AIRO anchors the risk-orientated constructs that are central to assurance under AI risk management guidance [3]. In combination, the vocabulary layer allows conformance matrices to reference controls and evidence using shared semantics, improving interoperability across tools and facilitating "multi-regime by design" instantiations.

### *5.2. Conformance-Matrix Metamodel and Mapping Rules*

The vocabulary layer provides semantic interoperability, but audit readiness additionally requires a *structural* representation of how requirements, controls, artefacts, and evidence relate. We therefore define a conformance-matrix metamodel that specifies the entities, identifiers, and relations required to produce *audit-ready conformance matrices* as first-class operational artefacts.

Core Entities

The metamodel is centred on the following entity families:

- **Normative Requirement** (NR): an atomic requirement statement derived from a selected regime (e.g., GDPR; EU AI Act), expressed with a unique identifier and explicit scope constraints [1,2].
- **Control Objective** (CO): a verifiable objective that operationalises one or more NR items in a technical setting, aligned with assurance-orientated practice [3].
- **Control** (C): an implementable control specification (policy, process, technical mechanism, or check) that realises a CO.
- **Evidence Specification** (ES): a definition of the minimal evidence required to assess a control (evidence type, provenance expectations, responsible role, retention, and acceptable variance).
- **Pipeline Artefact Type** (AT) and **Evidence Item** (EI): respectively, a class of artefacts produced by the pipeline (e.g., dataset lineage record; model card; risk register entry) and a concrete evidence item generated for a given run or review cycle.
- **Gate** (G) and **Decision Record** (DR): an explicit checkpoint that evaluates one or more ES items and records pass/fail decisions with rationale and accountability metadata.

To support auditability, all entities use stable identifiers and typed relations. Provenance for EI and DR is captured using established provenance modelling patterns to enable reproducible audit queries (e.g., who generated an artefact, when, using which inputs) [8].

Table 3 provides a compact view of the conformance-matrix metamodel, including the core entity types, suggested identifier patterns, and the trace relations that enable deterministic audit queries.

The resulting norm-to-evidence chain is explicit, testable, and stable under common lifecycle changes.

 doi:10.20944/preprints202601.1812.v1



**Table 3.** Conformance-matrix metamodel (at-a-glance): core entities, suggested identifiers, and key relations.

| Type | Role in the model | Suggested ID pattern | Key trace relations |
|---|---|---|---|
| NR | Atomic normative requirement (regime-specific, scoped) | `NR-<REG>-<NNN>` | NR → CO |
| CO | Verifiable control objective operationalising one/more NR items | `CO-<DOM>-<NNN>` | CO → C; CO → NR |
| C | Implementable control specification (policy/process/technical check) | `CTRL-<REG>-<DOM>-<NNN>` | C → ES; C → AT/EI |
| ES | Evidence specification (minimum evidence, checks, acceptance criteria) | `EVD-<DOM>-<NNN>` | ES → EI; ES → G |
| AT / EI | Artefact type / concrete evidence item (run-scoped instance) | `AT-<DOM>-<NNN>` / `EI-<RUN>-<NNN>` | EI @ RUN → DR; EI ↔ provenance links |
| G / DR | Gate and decision record (pass/fail + rationale + accountability) | `GATE-<STAGE>-<NNN>` / `DR-<RUN>-<NNN>` | G → DR; DR → (C, ES, RUN) |
| NC / CA | Nonconformity / corrective action (auditable remediation loop) | `NC-<RUN>-<NNN>` / `CA-<RUN>-<NNN>` | NC → CA; CA → (C, EI, DR) |

Trace Relations and Completeness Constraints

The metamodel encodes traceability as a bounded set of relations that enforce *norm-to-evidence* linkage: NR → CO → C → ES → (AT, EI) and ES → G → DR. This structure enables systematic coverage checks (e.g., requirements without controls; controls without evidence; evidence without provenance; gates without decision records) and makes audit readiness an operational property that can be tested rather than asserted. The traceability framing follows classic traceability principles, adapted to governance artefacts and evidence bundles [11].

Deterministic Mapping Rules

Beyond the entity set, the metamodel includes mapping rules that bind controls to concrete operational artefacts and evidence items. Each mapping rule is expressed as a deterministic constraint that specifies:

- which pipeline stage(s) a control applies to (e.g., data acquisition, training, evaluation, deployment, monitoring);
- which artefact types are required to evidence the control at that stage (AT);
- which minimal evidence items must be produced (EI), with provenance expectations;
- which gate(s) evaluate the evidence, and what decision records must be produced (DR).

These rules operationalise the compliance pipeline by making evidence requirements explicit and checkable. They also support versioning: when a requirement, control, or evidence specification changes, the impacted mappings can be identified, and the conformance matrix instance can be regenerated or updated without reworking unrelated parts.

### *5.3. Case-by-Case Instantiation Workflow*

The approach is *multi-regime by design*: the same metamodel can yield different conformance matrix instances depending on the selected regimes and the operational context. We therefore define a case-by-case instantiation workflow that produces a tailored, audit-ready matrix for a specific pipeline and deployment context.

Step 1: Scope the Context and Select Regimes

Define the operational context (system purpose, stakeholders, data categories, deployment constraints, and risk assumptions) and select the applicable regimes and internal policies to be instantiated. In this paper, the primary instantiation targets the **General Data Protection Regulation (GDPR)** and the **European Union Artificial Intelligence Act (EU AI Act)** [1,2].

 doi:10.20944/preprints202601.1812.v1



Step 2: Derive Atomic Normative Requirements

Decompose the selected regimes into a set of atomic NR items with unambiguous scope constraints (who/what/when applies). Record interpretive assumptions that affect implementation so that downstream controls and evidence can be evaluated against a stable interpretation baseline.

Step 3: Define Control Objectives and Controls

Translate NR items into CO items and implementable controls, favouring verifiability and operational feasibility. Each control is assigned (i) responsibility, (ii) frequency (per run; per release; periodic), and (iii) expected failure modes to support nonconformity handling in later stages.

Step 4: Specify Evidence and Gates

For each control, define ES (minimal evidence set, provenance expectations, retention, and acceptance criteria) and associate ES with one or more gates. Gates generate decision records that capture outcomes and rationales, enabling audit queries over pass/fail histories and supporting corrective action loops.

Step 5: Bind Controls to Pipeline Artefacts and Generate the Matrix Instance

Bind controls and evidence specifications to pipeline stages and artefact types via the mapping rules, generating a conformance matrix instance with stable identifiers and trace links. The resulting matrix is an operational artefact that can be versioned, queried, and reviewed alongside the technical pipeline.

To support reproducibility and consistent instantiation across contexts, Table 4 summarises the workflow as a sequence of steps with explicit inputs, outputs, and minimal quality checks.

**Table 4.** Case-by-case instantiation workflow: steps, inputs, outputs, and minimal quality checks.

| Step | Inputs | Outputs | Minimal quality checks |
|---|---|---|---|
| 1 | Context scope (system boundary, data flows, intended use) + selected regimes | Scope specification + assumptions baseline | Scope completeness; explicit assumptions recorded; regime versions fixed |
| 2 | Regime texts + scope specification | Atomic NR set (scoped requirement statements) | Atomicity; non-duplication; applicability conditions explicit |
| 3 | NR set + organisational policies/practices | CO set + control register (C) | Verifiability of CO/C; responsibility and frequency assigned |
| 4 | Controls (C) + lifecycle touchpoints | Evidence specifications (ES) + gate definitions (G) | Evidence sufficiency; acceptance criteria defined; gate coverage over lifecycle events |
| 5 | (NR, CO, C, ES, G) + mapping rules | Matrix instance + run-scoped evidence requirements | Trace completeness (NR→...→EI/DR); stable IDs; consistency constraints |

This workflow operationalises "multi-regime by design" by keeping the metamodel stable while varying regime mappings and context assumptions.

Conceptual Validation

The workflow is conceptually validated by (i) competency questions that test whether key audit questions can be answered from the model, (ii) consistency checks over trace completeness, and (iii) an illustrative instantiation of an AI risk pipeline, demonstrating how regime-specific requirements yield context-specific conformance matrix instances.

 doi:10.20944/preprints202601.1812.v1



## 6. Evidence Model and Audit Queries

The compliance pipeline is only as strong as the evidence it can *produce*, *verify*, and *retrieve* on demand. In practice, audit readiness often fails not because documentation is absent, but because trace links between normative requirements, operational controls, and run-level artefacts are fragmented, inconsistent, and costly to validate. To avoid retroactive, ad hoc reconstruction, we define an evidence model that is (i) *control-centric*, (ii) *run-aware*, (iii) *queryable*, and (iv) integrity-anchored. Evidence is specified *per control* (what must exist and what must be checked), collected *per run* (what was actually produced in a concrete execution of the technical pipeline), and validated through gates that condition promotion decisions (e.g., deployment) on explicit acceptance criteria. This section formalises (a) evidence bundles as first-class artefacts, (b) run-level traceability and integrity anchors, and (c) audit queries as operational primitives.

### 6.1. Evidence Bundles: Structure and Minimum fields

We define an *evidence bundle* as a structured package of artefacts and verification results that jointly satisfy the evidence specification of one or more controls for a given system instance and run. Evidence bundles are intentionally artefact-agnostic: they may include structured documentation (e.g., *model cards* and *datasheets*) [18,19], configuration snapshots (e.g., deployment manifests and access-policy exports), evaluation and monitoring reports, risk registers, and audit-relevant logs. The key requirement is that each artefact instance is linked to the control(s) it supports and is accompanied by check results that can be re-verified.

At the metamodel level (Section 5.2), each control is linked to an `EvidenceSpec` that declares: (i) required artefact types, (ii) minimum fields for each artefact instance, (iii) verification checks (syntactic and semantic), and (iv) acceptance criteria that drive gates. At the operational level, the evidence store records concrete `artefactInstance` objects together with `CheckResult` objects. This pairing provides audit-ready traceability between requirements, controls, and run-level evidence and supports accountability by making the *producer* and *verifier* explicit [4].

Table 5 summarises the minimum fields that enable reproducibility, traceability, and integrity checks without committing to a particular storage technology. Where needed, provenance links (agent/activity/entity) can be represented using standard provenance concepts [8]. The same bundle structure supports iteration: a detected nonconformity triggers a corrective action that produces new artefacts, re-runs check, and yields a new bundle version that supersedes (but does not erase) the prior state.

**Table 5.** Minimum fields for evidence bundles (control-centric and run-aware).

| Field | Purpose |
|---|---|
| `bundle_id` | Stable identifier for the evidence bundle. |
| `control_id` / `requirement_id` | Trace to the control(s) and normative requirement(s) the bundle supports. |
| `system_instance_id` | Binds evidence to a specific system/pipeline instance and operational context. |
| `run_id` | Links evidence to a concrete execution (training, evaluation, deployment, monitoring). |
| `artefact_id`, `artefact_type` | Identifies each artefact instance and its declared type (as per `EvidenceSpec`). |
| `location / pointer` | Reference to the artefact payload (file path, object-store key, database pointer). |
| `hash / checksum` | Integrity anchor for artefact payloads (tamper-evident verification). |
| `timestamp` | When the artefact/check was produced or last validated. |
| `producer_role / agent` | Who produced the artefact (role/agent), enabling accountability. |
| `check_id`, `check_result` | Verification results and statuses against acceptance criteria. |
| `status` | Aggregate bundle status (e.g., complete/incomplete; pass/fail; pending review). |

### 6.2. Run-Level Traceability and Integrity

Run-level traceability operationalises the distinction between *what is specified* (controls and evidence requirements) and *what actually happened* in the technical pipeline. A *run* is treated as an atomic,

 doi:10.20944/preprints202601.1812.v1



timestamped execution context that produces artefacts (or updates them) and yields check results. Typical run categories include: data ingestion and pre-processing, training, evaluation, packaging and deployment, monitoring, and incident response (including rollback). For each run, the evidence model captures the run category, input references (datasets, policies, configurations), output references (models, reports, logs), and the checks executed for each relevant control.

This run-aware design responds to well-known engineering realities in machine learning systems: system behaviour depends on data pipelines, configuration, and operational feedback loops, which can accumulate technical debt and hinder post-hoc analysis [5,6]. By anchoring evidence to runs, the compliance pipeline can detect when a change in data, model, or deployment invalidates previously acceptable evidence and therefore requires re-verification before gate promotion. In particular, evidence bundles can be marked as stale when their supporting artefacts are superseded by a new run, allowing auditors to reason explicitly about versioning, drift, and change impact.

Integrity is addressed by combining (i) stable identifiers, (ii) content-based hashes for artefact payloads, and (iii) provenance links that represent who/what produced artefacts and when [8]. The goal is not to prescribe a single security mechanism but to ensure that auditors can verify completeness and detect inconsistencies across runs (e.g., a model promoted to production without the minimum evidence for a required control). This approach is consistent with governance-by-design practices and supports risk management workflows that require continuous monitoring and documentation of controls across the lifecycle [3].

### *6.3. Audit Queries and Competency Questions*

To be operationally useful, the matrix and evidence store must support audit queries that answer concrete *competency questions* about compliance status, coverage, and corrective actions. Audit queries are defined over the trace graph induced by the matrix metamodel: `Requirement` → `Control` → `EvidenceSpec` → `ArtefactInstance/CheckResult @ Run` → `Gate/Status` → `Nonconformity/CorrectiveAction`. This structure supports the systematic retrieval of evidence and enables auditors to cross-check whether acceptance criteria were met at the time of a gate decision.

Examples of audit queries include:

- **Coverage and completeness:** Which controls mapped from GDPR and the EU AI Act are currently uncovered or have missing/failed evidence for the latest approved run?
- **Gate compliance:** For a given deployment, which gates were passed, with which evidence bundles, and under which acceptance criteria?
- **Lineage and change impact:** For a given requirement, which artefacts and runs support it, and what changed between two approved runs?
- **Nonconformities:** Which nonconformities are open, which controls they affect, and which corrective actions are pending or verified?
- **Risk-to-control trace:** For a given AI risk concept (AIRO-aligned), which mitigation controls are instantiated and what evidence exists for their implementation and verification?

These queries provide the operational backbone for audit readiness: they enable fast retrieval of evidence, consistency checks, and prioritisation of remediation. They also address practical audit limitations documented in the literature, where the availability and structure of evidence significantly affect audit validity and efficiency, and where black-box access is often insufficient for rigorous audit conclusions [13,15].

## 7. Implementation

This section provides a pragmatic implementation blueprint for representing the proposed artefacts in standard software engineering workflows while preserving run-level traceability. The design goal is conservative: the *compliance pipeline* should (i) maintain a versioned conformance-matrix *template* and its context-specific *instantiations*, (ii) collect run-scoped evidence bundles with integrity metadata, and (iii) enable deterministic audit retrieval and validation. The blueprint is intentionally

 doi:10.20944/preprints202601.1812.v1



agnostic to particular MLOps stacks, storage backends, and ontology tooling; it follows the practical observation that, without stable identifiers and disciplined versioning, ML systems tend to accumulate operational complexity and technical debt under iterative changes. [5,6]

### *7.1. Representations and Repository Structure*

A minimal representation can be achieved with a small set of machine-readable formats:

- **Ontologies and vocabularies:** OWL/RDF serialisations (e.g., Turtle) for the adopted vocabularies (DPV, AIRO) and any project-specific extensions, each with an explicit version identifier.
- **Conformance-matrix template and instantiations:** (a) a human-auditable tabular view (CSV) and (b) a typed machine-readable view (YAML/JSON). The tabular view supports review and sampling; the structured view enables unambiguous typing and automated validation.
- **Evidence bundles:** a per-run bundle (JSON/YAML) captURIsng referenced artefact instances (paths/URIs), verification results, and integrity metadata (hashes, timestamps, tool versions).
- **Audit query pack:** a simple specification (YAML/JSON) that encodes competency questions, parameters (e.g., control ID, run ID, time window), and retrieval scopes.

An illustrative repository layout that supports versioning and run-level retrieval is shown below:

```
repo/
  normative_sources/              # GDPR, EU AI Act, and optional extensions (versioned)
  ontologies/                     # DPV, AIRO, project extensions (versioned)
  matrix_templates/               # metamodel schema + template matrices
  instantiations/<context_id>/    # instantiated matrices, control register, mapping tables
  audit_queries/                  # query pack (competency questions)
  runs/<run_id>/                  # run-level evidence bundles and referenced artefacts
    evidence_bundle.json
    artefacts/
    checks/
    gates/
    nonconformities/
```

Within `runs/<run_id>/`, referenced artefacts SHOULD be treated as immutable once recorded (append-only strategy), with integrity captured via content hashes. If an artefact must be updated, it SHOULD be recorded as a new artefact instance linked to the original via a trace relation and accompanied by a corrective action record. This approach reduces post hoc reconstruction effort and preserves stable trace links across runs. [6]

### *7.2. Evidence Bundle Schema and Identifiers*

To support deterministic audit retrieval, each *control* and *evidence specification* SHOULD have a stable identifier and a minimal set of required fields. A simple naming convention is sufficient, for example `CTRL-<regime>-<domain>-<number>` and `EVD-<domain>-<number>`. The evidence bundle then records, for each evidence specification, which concrete artefact instances were used, which checks were executed, and which gate decision was reached.

An illustrative evidence bundle (minimal fields) is shown below:

```
{
  "run_id": "RUN-2026-01-21-001",
  "context_id": "CTX-AIS-RISK-01",
  "template_version": "TEMPLATE-1.0.0",
  "instantiation_version": "INST-CTX-AIS-RISK-01-1.0.0",
  "evidence_items": [
    {
      "evidence_spec_id": "EVD-DATA-001",
      "control_ids": ["CTRL-GDPR-DATA-001"],
      "artefact_refs": ["runs/RUN-.../artefacts/dataset_lineage.json"],
```

 doi:10.20944/preprints202601.1812.v1



```
      "checks": [{"check_id": "CHK-HASH-001", "result": "pass"}],
      "integrity": {"sha256": "<hex>", "timestamp_utc": "2026-01-21T10:15:00Z"}
    }
  ],
  "gates": [
    {"gate_id": "GATE-PROMOTE-001", "result": "pass",
     "decision_record": "runs/RUN-.../gates/decision.json"}
  ]
}
```

This structure supports the audit queries described in Section 6: coverage (which controls lack evidence), verification status (which checks passed/failed), and gate compliance (which promotions were allowed and why). Where deeper lineage is required, evidence items can be augmented with W3C PROV-aligned relations (entity/activity/agent) to provide interoperable provenance across tools. [8]

### *7.3. Matrix Templates, Instantiation, and Integration Hooks*

The implementation distinguishes (i) a *template layer* (regime-agnostic metamodel schema plus default mapping rules) from (ii) *instantiations* (context-scoped controls, evidence specifications, and gate policies). Instantiation is operationalised as a deterministic transformation: given a scope specification (selected regimes, system context, scoping assumptions) and a versioned template, the output is an instantiated matrix plus the evidence specifications required for gating. During execution, the compliance pipeline validates whether the run-level evidence satisfies these specifications and produces a status vector per control (e.g., `pass/fail/incomplete`) that can be aggregated into gate decisions.

Integration into a conventional machine learning workflow can be achieved conservatively through four instrumentation hooks: data preparation, training/validation, deployment, and monitoring. At each hook, the technical pipeline emits artefacts (e.g., dataset lineage metadata, training configuration, evaluation reports, deployment manifests) that the compliance pipeline references as evidence instances. Documentation artefacts such as *model cards* and *datasheets* can be included when available or required by organisational policy, but are treated as evidence artefacts bound to controls rather than as the narrative centre of compliance. [18,19]

Gates can be implemented as policy checks in the same automation layer that promotes models to the next lifecycle stage (e.g., continuous integration/continuous deployment jobs or release approval steps). Importantly, the compliance pipeline does not require a specific tool stack: the minimal requirement is the ability to (i) assign run identifiers, (ii) store immutable evidence references with integrity metadata, and (iii) record gate outcomes and corrective actions in an append-only log.

## 8. Evaluation

This article is deliberately foundational: it specifies a dual-flow compliance architecture, an ontology-aligned vocabulary layer, a conformance-matrix metamodel, and an instantiation workflow that together enable *operational audit readiness* through verifiable, retrievable evidence. Accordingly, the evaluation focuses on *design validity* and *audit utility* (assurance-oriented properties) rather than on predictive performance. This stance is consistent with internal algorithmic auditing and assurance framings that emphasise evidence-backed accountability structures over narrative reporting. [4,14]

### *8.1. Evaluation Objectives and Protocol*

We evaluate the proposal with an artefact- and query-driven protocol aligned with the paper's primary instantiation under the *General Data Protection Regulation* (GDPR) and the *EU Artificial Intelligence Act* (EU AI Act), while avoiding any claim of exhaustive legal coverage. The protocol operationalises three objectives:

- **O1 (Metamodel sufficiency):** the metamodel includes the entities and relations required to represent a complete *norm-to-evidence* trace chain (requirement → control objective → con-

 doi:10.20944/preprints202601.1812.v1



trol → evidence specification → evidence items), including gates, decision records, and nonconformity/corrective-action handling.

- **O2 (Trace-link verifiability):** trace links are typed, directionally meaningful, and stable under common change events (e.g., new model versions), and provenance can be expressed in an interoperable form. [8,11]
- **O3 (Audit-query readiness):** the artefacts enable deterministic answers to core audit questions (coverage, completeness, verification status, and gate outcomes) using the audit queries defined in Section 6. [3]

These objectives are assessed through four evaluation steps:

1. **Structural inspection:** verify that the metamodel provides explicit entities for *normative source*, *requirement*, *control*, *evidence specification*, *check*, *gate*, *run*, *decision record*, *nonconformity*, and *corrective action*, enabling end-to-end traceability.
2. **Consistency checks:** validate that the vocabulary layer and the metamodel do not introduce contradictory typing or relation constraints, and that identifiers are unambiguous and stable (e.g., one evidence specification cannot satisfy incompatible control expectations without an explicit scope rationale).
3. **Audit exercises:** execute representative audit queries over an illustrative instantiation (AI risk pipeline) to retrieve control coverage, evidence completeness, verification outcomes, and gate decisions. This targets audit demands that are not reliably supported by black-box-only access. [15]
4. **Change-impact exercise:** apply representative changes (dataset update, feature pipeline change, model retraining) and verify that impacted controls and evidence specifications can be identified and re-verified, addressing trace erosion and hidden technical debt commonly observed in machine-learning systems. [5,6]

### *8.2. Competency Questions and Audit Tasks*

Following common validation practice for knowledge models, we use competency questions to test whether the proposed structures support the intended audit tasks. Each question maps to an explicit traversal pattern over the metamodel (Section 5) and to one or more audit queries (Section 6). The competency questions used in this paper are:

1. **CQ1 (Scope and rationale):** Which GDPR/EU AI Act-derived controls are in-scope for a given system context, which are out-of-scope, and what scoping assumptions justify this decision?
2. **CQ2 (Evidence specification):** For a given control, what evidence is minimally required, which artefact types can satisfy it, and which checks must pass?
3. **CQ3 (Run-level status):** For a selected run, which controls are *pass*/*fail*/*incomplete*, which evidence items support each status, and which gate decisions followed?
4. **CQ4 (Nonconformities and corrective actions):** Which nonconformities are open, which controls they affect, which corrective actions were prescribed, and whether re-verification closed them?
5. **CQ5 (Change impact):** Given a change in data, features, or model version, which controls and evidence specifications must be re-verified before promotion can proceed?

Collectively, these questions operationalise the central claim of the paper: audit readiness depends on maintaining an evidence-backed, queryable trace structure across the lifecycle, not on producing documentation *post hoc*. They also align with assurance-orientated governance expectations that treat coverage, completeness, and verification status as assessable properties. [3,4]

### *8.3. Threats to Validity and Limitations*

This evaluation is conservative and acknowledges three limitations. First, the paper does not report large-scale empirical measurements (e.g., time-to-audit across many projects); its goal is to establish a reusable structure for auditability that can later be evaluated quantitatively in broader deployments. Second, although the GDPR and the EU AI Act provide the primary instantiation, regulatory interpretation and organisational policy vary; each instantiation must document assumptions,

 doi:10.20944/preprints202601.1812.v1



scope boundaries, and mappings from requirements to controls and evidence specifications. Third, the illustrative instantiation demonstrates feasibility but does not claim completeness: rigorous coverage of all obligations is inherently context-specific and may require legal review and organisational process alignment.

Within these constraints, the evaluation supports the main design claim: the proposed metamodel and trace-link scheme are sufficient to (i) express controls as operational evidence expectations, (ii) retrieve run-level evidence and verification status through audit queries, (iii) enforce promotion gates based on evidence, and (iv) manage nonconformities and corrective actions in a way that remains stable under system evolution.

## 9. Discussion

### *9.1. Operational Audit Readiness as an Engineering Property*

This work argues that *operational audit readiness* should be treated as an engineering property of AI-enabled decision-support systems, rather than as a documentation activity performed after deployment. In high-stakes contexts, assurance claims must be supported by evidence that can be located, checked, and replayed against explicit acceptance criteria. Prior work has highlighted persistent accountability and auditability gaps when evidence remains dispersed across heterogeneous artefacts and organisational narratives [4,13]. The proposed dual-flow architecture addresses this gap by making a compliance pipeline a first-class operational component: it registers controls, specifies evidence requirements, binds check to gates, and records decisions as structured artefacts. This framing also complements established observations from software engineering for machine learning that technical pipelines accrue "hidden" maintenance costs and verification friction when governance requirements are retrofitted [5,6].

A second implication is that audit readiness cannot be achieved by access alone. Even when auditors are given substantial access, meaningful verification requires stable identifiers, trace links, and checkable criteria that connect normative requirements to specific artefacts and evidence items [15]. Accordingly, the conformance-matrix metamodel is positioned as the minimal structure that makes verification queries deterministic: requirements and controls are mapped to evidence specifications, and evidence specifications are bound to artefact types, integrity signals, and gate decision records.

### *9.2. Interoperability and Multi-Regime Generality*

A central design choice is to separate (i) a reusable metamodel for conformance matrices and trace links from (ii) regime-specific instantiations of requirements and controls. This enables a *multi-regime by design* approach: a new regime is incorporated by adding a requirements-to-controls mapping and corresponding evidence specifications while preserving the same artefact, gate, and decision-record primitives. Interoperability is strengthened by aligning the vocabulary with established semantic resources for privacy and AI risk, and by using provenance constructs to support consistent evidence attribution [8,9,12]. This alignment supports cross-organisational portability: different implementations can share control semantics and audit-query patterns even when their underlying technical stacks differ.

### *9.3. Limitations and Future Work*

The contribution is deliberately foundational. First, the paper does not claim normative completeness for the General Data Protection Regulation (GDPR) or the European Union Artificial Intelligence Act (EU AI Act); it provides an instantiation pattern that can be extended as organisational obligations, guidance, and enforcement expectations evolve [1,2]. Second, the approach assumes a minimum level of process discipline (stable identifiers, artefact versioning, and retention policies) that may be difficult to sustain in organisations with weak governance maturity. Third, the work focusses on traceability and verifiability rather than on security guarantees for evidence storage and access control, which must be specified and assessed as part of deployment-orientated engineering and risk management [3].

 doi:10.20944/preprints202601.1812.v1



Future work should prioritise: (i) empirical evaluations in operational environments to quantify verification effort, failure modes, and change-impact costs; (ii) reference tooling that generates evidence bundles and gate decision records with integrity protections; and (iii) curated libraries of audit queries and evidence specifications that integrate common artefacts such as model cards and datasheets as control-scoped evidence, rather than as stand-alone compliance narratives [18,19].

## 10. Conclusions

Operational audit readiness for high-stakes AI decision-support systems is increasingly assessed as a lifecycle capability: organisations must be able to demonstrate, with retrievable and checkable evidence, that governance requirements were implemented, verified, and maintained under continuous change. In practice, this capability often breaks down not because documentation is missing, but because trace links from normative obligations to operational controls and concrete pipeline artefacts are incomplete, inconsistent, or costly to reconstruct. This paper addresses that gap by establishing a foundation for *audit-ready conformance matrices* using a **dual-flow, layered architecture** that separates (i) a **conventional** technical ML pipeline from (ii) a **novel** compliance pipeline engineered to operationalise governance requirements as verifiable artefacts and decisions.

The technical flow is intentionally treated as a baseline execution context organised into five layers (**Data**, **Feature**, **Model**, **Deployment**, **Monitor**). These layers reflect established MLOps practices and the known reality that models, data, and configurations evolve continuously. The main contribution lies in the second flow: a compliance pipeline structured into five layers (**Scope**, **Requirements**, **Controls**, **Evidence**, **Assurance**) and coupled to technical lifecycle events through an explicit **Traceability Layer**. This interface is designed to propagate stable identifiers (notably `run_id`), bind controls and evidence specifications to concrete pipeline artefacts, and support deterministic retrieval of *run-level evidence bundles* aligned with gate decisions and decision records.

Within this framing, the paper delivered a coherent set of reusable artefacts that together make the norm-to-evidence chain explicit and testable. First, an **ontology-aligned vocabulary layer** leverages the **Data Privacy Vocabulary (DPV)** and the **AI Risk Ontology (AIRO)** to reduce semantic drift across regimes, stakeholders, and artefact types. Second, a **conformance-matrix metamodel** that defines core entities and typed relations (NR, CO, C, ES, EI, G, DR, NC/CA) and supports bounded trace chains of the form NR $\rightarrow$ CO $\rightarrow$ C $\rightarrow$ ES $\rightarrow$ EI @ RUN $\rightarrow$ (G $\rightarrow$ DR), with explicit nonconformity and corrective-action handling. Third, **deterministic mapping rules** that bind controls and evidence specifications to technical touchpoints and artefact types, enabling completeness checks (e.g., controls without evidence, gates without decision records) and change-impact identification. Fourth, a **case-by-case instantiation workflow** that produces context-scoped conformance matrix instances with explicit inputs/outputs and minimal quality checks, supporting *multi-regime by design* while preserving a stable core structure. Fifth, an **evidence model and audit-query layer** that treats evidence as (i) control-centric, (ii) run-aware, and (iii) integrity-anchored, enabling operational queries over coverage, completeness, gate compliance, lineage/change impact, and remediation status.

To demonstrate feasibility without over-claiming, the paper provided a primary instantiation for the **General Data Protection Regulation (GDPR)** and the **EU Artificial Intelligence Act (EU AI Act)**. This instantiation illustrates how normative anchors can be translated into control objectives, implementable controls, evidence specifications, and gate-verifiable decisions, while maintaining explicit scoping assumptions and stable identifiers. The validation is deliberately foundational and assurance-orientated: it emphasises design validity and audit utility via competency questions and consistency checks, rather than predictive performance or claims of normative completeness. Where explainability artefacts are used, they are treated only as *optional evidence items* associated with specific controls, not as the organising centre of the compliance pipeline.

Three limitations follow directly from this scope. First, instantiation remains context-dependent: regime interpretation, organisational policy, and system boundaries must be stated explicitly and versioned to preserve audit meaning. Second, operational audit readiness assumes baseline process

 doi:10.20944/preprints202601.1812.v1



discipline (identifier stability, artefact versioning, retention, and append-only decision records); without it, trace structures erode under drift. Third, while integrity anchoring is specified at the evidence level (e.g., hashes/checksums and provenance-aligned attribution), deployment-specific security controls for evidence storage and access must be addressed within the system's risk management programme.

Future work should proceed along three practical lines. (i) **Empirical evaluation** in operational settings to quantify verification effort, failure modes, and change-impact costs relative to post-hoc reconstruction. (ii) **Reference tooling** that automates evidence bundle generation, trace-link validation, and gate decision records with minimal integration burden across common MLOps stacks. (iii) **Reusable libraries** of mappings, evidence specifications, and audit queries for additional regimes and organisational controls, preserving the same meta-model primitives to support consistent multi-regime instantiation.

Overall, the paper reframes Compliance-by-Design as an operational property delivered by a dual-flow architecture: a conventional technical pipeline complemented by a compliance pipeline that produces audit-ready conformance matrices and run-level evidence bundles under explicit gates and decision records. **In this model, audits become queries over traceable evidence, not reconstructions from scattered artefacts.**

**Author Contributions:** Conceptualization, A.G. and A.C.; methodology, A.G.; software, A.G.; validation, A.G. and A.C.; formal analysis, A.C.; investigation, A.G.; resources, A.C.; writing—original draft preparation, A.G.; writing—review and editing, A.G. and A.C.; visualization, A.G.; supervision, A.C.; project administration, A.G. and A.C.; funding acquisition, A.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). Official Journal of the European Union, L 119, 2016.
2. European Parliament and Council of the European Union. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union, 2024.
3. National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). Technical Report NIST AI 100-1, U.S. Department of Commerce, 2023. https://doi.org/10.6028/NIST.AI.100-1.
4. Raji, I.D.; Smart, A.; White, R.N.; Mitchell, M.; Gebru, T.; Hutchinson, B.; Smith-Loud, J.; Theron, D.; Barnes, P. Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing. In Proceedings of the Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAccT '20). Association for Computing Machinery, 2020, pp. 33–44. https://doi.org/10.1145/3351095.3372873.
5. Amershi, S.; Begel, A.; Bird, C.; DeLine, R.; Gall, H.; Kamar, E.; Nagappan, N.; Nushi, B.; Zimmermann, T. Software Engineering for Machine Learning: A Case Study. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2019, pp. 291–300. https://doi.org/10.1109/ICSE-SEIP.2019.00042.
6. Sculley, D.; Holt, G.; Golovin, D.; Davydov, E.; Phillips, T.; Ebner, D.; Chaudhary, V.; Young, M.; Crespo, J.F.; Dennison, D. Hidden Technical Debt in Machine Learning Systems. In Proceedings of the Advances in Neural Information Processing Systems, 2015, Vol. 28, pp. 2503–2511.

7. Winkler, S.; von Pilgrim, J. A Survey of Traceability in Requirements Engineering and Model-Driven Development. *Software and Systems Modeling* **2010**, *9*, 529–565. https://doi.org/10.1007/s10270-009-0145-0.
8. Lebo, T.; Sahoo, S.; McGuinness, D. PROV-O: The PROV Ontology. W3C Recommendation, 2013.
9. W3C Data Privacy Vocabularies and Controls Community Group (DPVCG). Data Privacy Vocabulary (DPV) — Version 2.0. Specification, 2025.
10. Golpayegani, D.; Pandit, H.J.; Lewis, D. AIRO: An Ontology for Representing AI Risks Based on the Proposed EU AI Act and ISO Risk Management Standards. In Proceedings of the Proceedings of the 18th International Conference on Semantic Systems (SEMANTiCS 2022). IOS Press, 2022, pp. 101–108. https://doi.org/10.3233/SSW220008.
11. Gotel, O.; Finkelstein, A. An Analysis of the Requirements Traceability Problem. In Proceedings of the Proceedings of the First International Conference on Requirements Engineering. IEEE Computer Society, 1994, pp. 94–101. https://doi.org/10.1109/ICRE.1994.292398.
12. AI Risk Ontology (AIRO) Community. AIRO: AI Risk Ontology. Ontology specification, 2025.
13. Hartmann, D. Addressing the regulatory gap: moving towards an EU AI audit framework. *AI and Ethics* **2024**. https://doi.org/10.1007/s43681-024-00595-3.
14. Burr, C.; Leslie, D. Ethical assurance: a practical approach to the responsible design, development, and deployment of data-driven technologies. *AI and Ethics* **2023**, *3*, 73–98. https://doi.org/10.1007/s43681-022-00178-0.
15. Casper, S.; Ezell, C.; Siegmann, C.; Kolt, N.; Curtis, T.L.; Bucknall, B.; Haupt, A.A.; Wei, K.; Scheurer, J.; Hobbhahn, M.; et al. Black-Box Access is Insufficient for Rigorous AI Audits. In Proceedings of the Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24). Association for Computing Machinery, 2024, pp. 2254–2272. https://doi.org/10.1145/3630106.3659037.
16. Zarour, M.; Alzabut, H.; Al-Sarayreh, K.T. MLOps Best Practices, Challenges and Maturity Models: A Systematic Literature Review. *Information and Software Technology* **2025**, *183*, 107733. https://doi.org/10.1016/j.infsof.2025.107733.
17. Pandit, H.J.; Polleres, A.; Bos, B.; Brennan, R.; Bruegger, B.; Ekaputra, F.J.; Fernández, J.D.; Hamed, R.G.; Kiesling, E.; Lizar, M.; et al. Creating a Vocabulary for Data Privacy — The First-Year Report of Data Privacy Vocabularies and Controls Community Group (DPVCG). In Proceedings of the On the Move to Meaningful Internet Systems: OTM 2019 Conferences. Springer, 2019, Vol. 11877, *Lecture Notes in Computer Science*, pp. 714–730. https://doi.org/10.1007/978-3-030-33246-4_44.
18. Mitchell, M.; Wu, S.; Zaldivar, A.; Barnes, P.; Vasserman, L.; Hutchinson, B.; Spitzer, E.; Raji, I.D.; Gebru, T. Model Cards for Model Reporting. In Proceedings of the Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* '19). ACM, 2019, pp. 220–229. https://doi.org/10.1145/3287560.3287596.
19. Gebru, T.; Morgenstern, J.; Vecchione, B.; Vaughan, J.W.; Wallach, H.; Daumé III, H.; Crawford, K. Datasheets for Datasets. *Communications of the ACM* **2021**, *64*, 86–92. https://doi.org/10.1145/3458723.