9 April 2026

AEGIS: An Evidence-Obligation Framework for AI-Native Enterprise Architecture

Madhava Gaikwad[1] and Mohadeb Mondal[2]

[1] Independent Researcher, gaikwad.madhav@gmail.com
[2] Independent Researcher, mohadeb.mondal@gmail.com

Abstract. Enterprise architecture (EA) frameworks such as TOGAF and ArchiMate were designed for deterministic, human-driven systems. Their artefacts are static documents, their governance relies on periodic review, and their metamodels do not represent evidence, obligation, or runtime conformance. AI-native systems built from autonomous agents, large language models, and continuously retrained pipelines do not fit these assumptions. Recent work has produced useful artefacts at lower layers of AI governance: Policy Cards encode per-agent obligations; AI Bills of Materials track model provenance; Attestable Audits cryptographically verify benchmark claims; LLM Audit Trails log lifecycle events; and ArchiMate Next adds AI agents as first-class modelling elements. What is missing is a framework that integrates these artefacts into a continuous EA lifecycle that carries evidence. **AEGIS** (**A**rchitecture with **E**vidence, **G**overnance, **I**ntent, and **S**afety) is a typed ontology and lifecycle framework that: (i) introduces Evidence, Obligation, and RuntimeSignal as first-class EA objects; (ii) connects them to architectural entities through seven machine-checkable typed relations; (iii) lifts architectural fitness functions from code-level to EA-lifecycle scope; and (iv) crosswalks obligations to the EU AI Act, ISO/IEC 42001, and NIST AI RMF. Three expressiveness constructions evaluate the framework: mapping all nine ISO 42001 Annex A control domains to typed obligations, deriving obligation-drift detection from the metamodel's closure rule, and integrating Policy Cards into an EA-level conformance graph. Each construction captures a class of governance failure that existing tools cannot detect or represent.

Keywords: enterprise architecture · AI governance · evidence ontology · obligation semantics · continuous conformance · EU AI Act · ISO 42001 · large language models · multi-agent systems

1 Introduction

Enterprise architecture (EA) is the discipline that aligns an organisation's business strategy, information assets, applications, and technology infrastructure. The field is governed by two open standards: TOGAF [18] and ArchiMate [19]. Both were designed for deterministic, human-driven systems where components behave predictably, governance happens in Architecture Review Board meetings,

and an architecture document remains useful for months or years after it is written.

AI-native systems create governance problems outside this scope. A deployed large language model produces probabilistic outputs. An autonomous agent orchestrating tool calls may behave differently depending on context it has never seen before. A retrieval-augmented generation pipeline changes its effective knowledge base daily as documents are ingested. When such systems make consequential decisions about credit, hiring, patient routing, or infrastructure configuration, static diagrams are not enough to determine whether the system is behaving as specified or whether it remains within its regulatory obligations. That requires continuous, runtime, machine-checkable governance.

The regulatory pressure is concrete. The EU AI Act entered into force in August 2024 and imposes binding obligations: technical documentation, conformity assessment, post-market monitoring, and incident reporting, with high-risk system requirements enforceable from August 2026 [4]. ISO/IEC 42001:2023 requires a traceable evidence index linking each obligation to specific processes and artefacts [7]. NIST AI RMF identifies traceability as a core characteristic of trustworthy AI [1]. Enterprise architects are therefore expected to produce conformity evidence at a level of detail and at a cadence that current EA tools do not support.

Related artefacts address individual layers. ArchiMate Next (2025) adds AI agents as first-class modelling elements [20]. Architecture as Code [5] makes fitness functions executable via the Model Context Protocol (MCP) [2]. Policy Cards [13] encode per-agent obligations in machine-readable JSON Schema. AI-BOMs [3] track model supply-chain provenance. Attestable Audits [17] cryptographically verify benchmark claims. LLM Audit Trails [15] log lifecycle events tamper-evidently. The AIRO/TAIR ontology cluster [6,8] formalises EU AI Act requirements as Open Knowledge Graphs.

Each of these artefacts operates in isolation. A Policy Card governs one deployed agent. It does not know which architectural Service that agent implements, which DataAsset it accesses, which regulatory Obligation that access is subject to, or whether the last model update recorded in the LLM Audit Trail has been verified against the benchmark evidence in the conformity file. The Policy Cards authors identify this gap directly: no existing artefact connects agent-layer engineering reality to enterprise-level regulatory expectation [13].

This paper. AEGIS is a typed ontology and lifecycle framework at the EA layer above the existing artefacts. Three contributions:

1. **An EA evidence-obligation metamodel.** Evidence, Obligation, Policy, Risk, RuntimeSignal, and Exception are introduced as first-class architectural objects, connected to existing EA entities through seven typed relations. AEGIS extends ArchiMate Next's core concept set and aligns with AIRO/TAIR/DPV as a domain-specific profile.

2. **A continuous conformance lifecycle.** TOGAF's document-centric ADM is replaced with a six-phase loop: *Intent*, *Model*, *Verify*, *Deploy*, *Observe*, *Adapt* (IMVDOA). Architectural fitness functions are lifted from code-level to EA-lifecycle scope. Runtime signals from Policy Card enforcement, LLM Audit Trails, and Attestable Audit outputs are consumed as EVIDENCE objects through a PROV-O extension.
3. **A regulatory crosswalk.** All 38 ISO/IEC 42001 controls and the key EU AI Act Articles are mapped to obligation types in the metamodel, enabling the "Article X → Test Y → Evidence Z" triple required by AI management system (AIMS) conformity files.

Problem statement. Three governance failures motivate the work. *Evidence vacuum*: compliance claims exist only as text annotations; no typed, expiry-aware, machine-checkable link connects a claim to an artefact. *Obligation/architecture disconnect*: Policy Cards encode per-agent obligations, while regulations target systems. EU AI Act Art. 9 requires a risk management system across the full lifecycle, yet no current framework maps this requirement to architectural entities. *Obligation drift*: a model may be retrained without re-running its conformity assessments, or supporting evidence may expire without any signal reaching governance workflows. Either case constitutes non-conformance under ISO/IEC 42001 and the EU AI Act. No existing tool detects this class of failure.

Paper organisation. Section 2 surveys related work. Section 3 presents the AEGIS metamodel. Section 4 describes the IMVDOA lifecycle. Section 5 gives the regulatory crosswalk. Section 6 presents the expressiveness evaluation. Section 7 discusses limitations. Section 8 concludes.

2 Background and Related Work

Five properties are required for AI-native EA governance: typed evidence objects at EA scope; obligation semantics; a full EA lifecycle; a continuous runtime conformance loop; and LLM-native agent support with a regulatory crosswalk. Table 1 maps coverage across relevant traditions.

EA frameworks. TOGAF 10 [18] and ArchiMate 3.x [19] govern lifecycle and modelling respectively. Neither carries evidence or obligation semantics. ArchiMate Next [20] adds a `Role` element applicable to AI agents. It adds no evidence objects, obligation semantics, or runtime conformance loop. *Architecture as Code* [5] makes fitness functions executable via MCP [2]. It lacks a regulatory crosswalk and an EA-level evidence ontology.

Regulatory ontologies and audit artefacts. AIRO [6] and TAIR [8] provide OWL2 classes for EU AI Act risk assessment. Neither includes architectural objects or a runtime loop. ISACA guidance [7] identifies the required conformity triple "Article X → Test Y → Evidence Z". It does not provide a machine-readable

Table 1. Coverage of five key properties across related work. ✓ = fully addressed; ~ = partially addressed; × = not addressed.

Tradition / Work	*Evidence objects*	*Obligation semantics*	*EA lifecycle*	*Runtime loop*	*LLM agents + reg.*
TOGAF [18] / ArchiMate [19]	×	×	✓	×	×
ArchiMate Next [20]	×	×	✓	×	~
Architecture as Code [5]	×	×	~	✓	~
AIRO / TAIR / DPV [6,8,16]	✓	✓	×	×	~
Policy Cards [13]	~	✓	×	✓	✓
LLM Audit Trails [15]	~	×	×	✓	✓
Attestable Audits [17]	✓	×	×	×	✓
Accountability Fabric [12]	✓	~	~	×	×
DDL Normative Supervisors [14]	×	✓	×	✓	×
AEGIS (this work)	✓	✓	✓	✓	✓

ontology for that triple. LLM Audit Trails [15] log lifecycle events, but they do not carry obligation semantics. Attestable Audits [17] sign benchmark results at the model layer. W3C PROV-O [10] provides provenance vocabulary, but it has not been extended to EA objects or obligation semantics.

Agent-layer governance. Policy Cards [13] encode per-agent obligations with EU AI Act and ISO 42001 crosswalk mappings validated via OPA/Rego (Open Policy Agent). No existing artefact connects agent-layer engineering to enterprise-level regulatory expectation [13]. The Accountability Fabric [12] proposes typed accountability objects and partial obligation semantics. It predates LLM agents and carries no regulatory crosswalk. Governatori's Defeasible Deontic Logic (DDL) [14] provides normative supervisors for individual agents. AEGIS adopts DDL for obligation evaluation at EA scope.

Table 1 shows that no existing work simultaneously satisfies all five properties.

3 The AEGIS Metamodel

3.1 Design Principles

AEGIS is built on four principles. (P1) Extension, not replacement: AEGIS profiles ArchiMate Next and extends PROV-O, so existing ArchiMate models require no re-modelling. (P2) Evidence is a first-class type with provenance, expiry, and status. (P3) Obligations are machine-checkable typed objects grounded in DDL

semantics. (P4) Conformance is continuous through a runtime loop rather than periodic governance gates.

3.2 Core Entity Types

AEGIS introduces seven new typed entities. These are defined as OWL2 classes and serialised in Turtle/JSON-LD; a full ontology is published at `[anonymised]`. Each entity type is aligned with PROV-O using the `prov:Entity`, `prov:Activity`, and `prov:Agent` mapping described in §3.

Evidence A typed, time-stamped, provenance-carrying artefact that attests a property of an architectural entity. Sub-types include operational artefacts (EvaluationReport, RedTeamResult, AttestableAuditResult, SLOReport), supply-chain artefacts (ModelCard, AIBOM, SystemCard), and regulatory compliance artefacts (InstructionsForUse per Art. 13; LoggingDesignSpec per Art. 12; QMSDocumentation per Art. 17; PostMarketMonitoring-Plan per Art. 72). Every instance carries `aegis:issuedBy`, `aegis:issuedAt`, `aegis:expiresAt`, `aegis:evidenceStatus` (`valid` | `expired` | `superseded` | `revoked`), and `aegis:strength` $\in [0, 1]$.

Obligation A machine-checkable normative requirement placed on an architectural entity. Each Obligation is associated with: a deontic modality (`must` | `must_not` | `should` | `may`); a trigger condition expressed in SHACL (Shapes Constraint Language, a W3C standard for validating RDF graphs) or OPA Rego (Open Policy Agent's policy language); one or more Evidence requirements that must be satisfied; a regulatory source (zero or more entries in the regulatory crosswalk, §5); and a priority.

Policy A named, versioned collection of Obligation instances scoped to a Service, DataAsset, or ArchitectureDomain; the EA-level counterpart to a Policy Card, carrying `aegis:policyVersion`, `aegis:effectiveDate`, and `aegis:supersedes`.

Risk A structured risk entry aligned with ISO 31000 and AIRO [6], with `aegis:likelihood`, `aegis:severity`, and `aegis:riskCategory`; linked to mitigating Obligation instances via `mitigated_by`.

RuntimeSignal An event emitted by a deployed system (Policy Card enforcement, LLM Audit Trail entry [15], Attestable Audit result [17], drift metric) that is promoted to an Evidence object once linked to an architectural entity and an Obligation.

Exception A time-bounded, machine-readable dispensation from an Obligation, recording `aegis:grantedBy`, `aegis:justification`, and `aegis:expiresAt`.

ArchitectureDecisionRecord An extension of the ADR concept [9] carrying the decision, alternatives, Obligation and Risk forces, and links to the Evidence that informed it. ADRs are machine-queryable.

3.3 Relation Vocabulary

AEGIS defines seven typed relations that connect the new entity types to each other and to existing ArchiMate entities (Table 2).

Table 2. AEGIS typed relations. *Domain* and *Range* use ArchiMate Next (AN) and AEGIS (A) namespace prefixes.

Relation	Domain	Range
`evidenced_by`	AN:ArchElement	A:Evidence
`constrained_by`	AN:ArchElement	A:Obligation
`verified_by`	A:Obligation	A:Evidence
`violates`	AN:ArchElement	A:Obligation
`mitigated_by`	A:Risk	A:Obligation
`supersedes`	A:Evidence	A:Evidence
`escalates_to`	A:RuntimeSignal	A:Exception ∪ AN:Actor

Formal semantics. Each relation is specified using Defeasible Deontic Logic (DDL) [14]. The key inference rule is the *evidence-obligation closure*:

$$\texttt{constrained\_by}(e, o) \land \neg\exists\, v.\, \texttt{verified\_by}(o, v) \land v.\texttt{status} = \texttt{valid} \implies \texttt{violates}(e, o) \tag{1}$$

Equation (1) states that any architectural entity constrained by an obligation violates that obligation when no valid evidence verifies it. This inference is evaluated continuously as part of the runtime loop (§4).

Alignment. AEGIS extends W3C PROV-O [10]: Evidence ⊑ `prov:Entity`, RuntimeSignal ⊑ `prov:Activity`, and `evidenced_by` maps to `prov:wasDerivedFrom` with `aegis:*` qualifications. AEGIS attaches to ArchiMate Next [20] by specialising `AN:ActiveStructure` to AI_Agent, `AN:PassiveStructure` to Model, and extending the `Motivation` aspect to include Obligation and Risk alongside Goal and Principle.

Running example. Service S1 is constrained by obligation O1 ("PII must not flow to an external endpoint unless a redaction proof exists", sourced from `eu_ai_act:Article10`). O1 is verified by evidence E1, an EvaluationReport with `expiresAt` = 2026-07-15. On 16 July 2026, E1's `evidenceStatus` transitions to `expired`. The closure rule (Eq. 1) fires: S1 now `violates` O1. The conformance loop raises an `ObligationViolation` RuntimeSignal and escalates it to the responsible actor. This example is used throughout §6.

4 The IMVDOA Lifecycle

AEGIS replaces TOGAF's linear ADM with a continuous six-phase loop: **I**ntent, **M**odel, **V**erify, **D**eploy, **O**bserve, **A**dapt (IMVDOA). *Intent* derives typed Obligation instances from regulatory sources (EU AI Act tier, ISO 42001 scope), organisational policies, and stakeholder constraints, consuming TAIR OKG [8] outputs to populate Annex III obligations automatically. *Model* attaches Policy,

AIBOM, and Obligation instances to every AI_Agent, Service, and DataAsset. *Verify* runs SHACL (Shapes Constraint Language) shapes to check structural completeness and a DDL theorem prover [14] to check obligation satisfiability; failures block deployment. *Deploy* injects or validates the corresponding Policy Card for each agent, registers a deployment event in the LLM Audit Trail [15], and activates fitness functions via MCP [2]. *Observe* consumes Policy Card enforcement events, audit trail entries, and Attestable Audit results [17] as RuntimeSignal instances, promoting verified signals to Evidence objects. *Adapt* is triggered when the per-entity conformance score $\kappa(e)$ falls below a threshold:

$$\kappa(e) = \frac{|\{o : \texttt{constrained\_by}(e, o),\ o.\texttt{deontic} = \texttt{must},\ \exists v.\, \texttt{verified\_by}(o, v),\ v.\texttt{status} = \texttt{valid}\}|}{|\{o : \texttt{constrained\_by}(e, o),\ o.\texttt{deontic} = \texttt{must}\}|} \tag{2}$$

Adapt re-runs the Intent/Model/Verify sub-cycle for the affected scope.

5 Regulatory Crosswalk

Every Obligation in AEGIS is linkable to one or more regulatory requirements. This enables the machine-readable conformity evidence required by ISO/IEC 42001 and the EU AI Act.

5.1 EU AI Act Mapping

Table 3 maps seven EU AI Act obligation clusters for high-risk systems to AEGIS Obligation types, building on the regulatory learning space analysis of Lewis et al. [11]. Article titles and numbers are verified against the Official Journal version of Regulation (EU) 2024/1689 (13 June 2024).

ISO/IEC 42001 mapping. ISO/IEC 42001 defines 38 controls across nine Annex A domains (A.2 to A.10). AEGIS maps each to a typed Obligation instance, enabling the conformity triple "Control X → Test Y → Evidence Z" required by a conformant AIMS [7] to be expressed as machine-queryable SPARQL over the obligation graph. Representative mappings: A.6.2.4 (verification) → `TestingObligation` → EvaluationReport; A.7.5 (data provenance) → `ProvenanceObligation` → AIBOM, DataLineage; A.6.2.8 (event logs) → `AuditTrailObligation` → LLMAuditTrail. This triple has no native representation in ArchiMate 3.2 or TOGAF.

6 Expressiveness Evaluation

Three *expressiveness constructions* evaluate the metamodel. Each is a structured demonstration that the metamodel can represent a class of governance artefact or detect a class of failure by logical inference alone. No deployed system is measured; the claims are about logical expressiveness. Empirical evaluation against production toolchains requires a full implementation and is left to future work.

Table 3. EU AI Act Article to AEGIS Obligation mapping (high-risk systems, Art. 6 & Annex III). Article titles and numbers verified against Regulation (EU) 2024/1689, OJ version 13 June 2024.

AI Act Article	Obligation type	Evidence type required
Art. 9 (Risk management system)	`RiskManagementObligation`	RiskAssessment
Art. 10 (Data and data governance)	`DataGovernanceObligation`	AIBOM, DataLineage
Art. 11 (Technical documentation)†	`TechnicalDocObligation`	ModelCard, SystemCard
Art. 12 / 19 (Record-keeping & auto. logs)‡	`LoggingObligation`	LoggingDesignSpec, LLMAuditTrail
Art. 13 (Transparency and info. to deployers)	`TransparencyObligation`	InstructionsForUse
Art. 17 (Quality management system)§	`QMSObligation`	QMSDocumentation
Art. 72 (Post-market monitoring)¶	`MonitoringObligation`	PostMarketMonitoringPlan, RuntimeSi

†A SystemCard is one implementation of Annex IV; full compliance may require additional fields. ‡Art. 12 is a design-time capability obligation (system must allow logging); Art. 19 is the operational keeper obligation (provider must retain logs). LoggingDesignSpec evidences Art. 12; LLMAuditTrail evidences Art. 19. §Art. 17 requires a product-level QMS; ISO/IEC 42001 governs organisational AI management at a different scope. ¶Art. 72(3) requires a documented plan as part of Annex IV (pre-deployment); RuntimeSignal(ongoing) evidences execution post-deployment.

6.1 E1: ISO/IEC 42001 Control Mapping

AEGIS maps all 38 ISO/IEC 42001 Annex A controls to typed Obligation instances with verified control numbers. The "Control X → Test Y → Evidence Z" conformity triple is expressed as machine-queryable SPARQL over the obligation graph. ArchiMate 3.2 and TOGAF have no native representation for this triple. Representative mappings: A.6.2.4 (verification) → `TestingObligation` → EvaluationReport; A.7.5 (data provenance) → `ProvenanceObligation` → AIBOM, DataLineage; A.6.2.8 (event logs) → `AuditTrailObligation` → LLMAuditTrail. The mapping is composable with the EU AI Act crosswalk (§5).

6.2 E2: Obligation Drift Detection by Logical Inference

Three classes of obligation drift are detectable by the evidence-obligation closure rule (1) over the AEGIS graph. ArchiMate 3.2 models and code-level fitness functions cannot detect any of them.

Class 1: Evidence expiry without re-verification. When an EvaluationReport instance's `expiresAt` timestamp passes, its `evidenceStatus` transitions to `expired`. The closure rule fires: $\texttt{constrained\_by}(e, o) \wedge \neg\exists v.\ \texttt{verified\_by}(o, v) \wedge v.\texttt{status} = \texttt{valid} \Rightarrow \texttt{violates}(e, o)$. This is a monotonic inference requiring only the expiry timestamp and the relation graph. No runtime measurement is needed. AEGIS detects the violation as soon as the timestamp passes.

Class 2: Policy Card / EA obligation mismatch. If a deployed Policy Card [13] is updated at the agent layer but the corresponding EA-level Policy object is not, the SHACL Verify phase detects a structural `PolicyMismatch` violation: the AI_Agent's `constrained_by` links reference an Obligation version whose `policyCardRef` no longer matches the Policy Card's `obligations` array. This is detectable at the next Verify cycle without runtime instrumentation.

Class 3: Regulatory update propagation failure. When the TAIR OKG [8] publishes a new version reflecting updated regulatory guidance, the AEGIS Intent phase re-derives the obligation set. Any Obligation instance whose `regulatorySource` link points to a superseded article version is flagged as a `PolicyMismatch` in the next Verify cycle. Detection latency is bounded by the TAIR OKG publication cadence, typically hours to days after regulatory publication.

All three classes are structural failures in the obligation graph, detectable by logical inference. ArchiMate has no Obligation or Evidence objects. Code-level fitness functions measure software quality, not regulatory conformance. Policy Card enforcers operate at agent scope, not EA scope.

6.3 E3: Policy Card Integration

When multiple Policy Cards govern agents implementing the same Service, their obligations may contradict each other or conflict with EA-level constraints. AEGIS lifts each card's `obligations` array to typed Obligation instances, attaches them to the relevant Service, crosswalks them to EU AI Act and ISO 42001 sources, and links them to Evidence objects with expiry dates. The DDL satisfiability checker [14] detects inter-card conflicts that agent-layer enforcement cannot see. The closure rule (1) fires when any evidence expires. The lifted graph adds architectural scope, a regulatory crosswalk, evidence expiry, cross-card conflict detection, and SPARQL queryability. None of these are expressible in Policy Cards alone.

7 Discussion and Limitations

The three constructions are expressiveness demonstrations. They do not report measurements from a deployed system. The crosswalk covers the EU AI Act and ISO/IEC 42001. Extension to other jurisdictions (GDPR, FDA AI/ML) requires additional tables. The DDL prover [14] has polynomial complexity. AEGIS has not been profiled at enterprise scale. A full implementation covering an OWL2 ontology, SHACL shapes, SPARQL closure queries, and a Policy Card lifting tool is the primary target for future work. Empirical evaluation against existing governance toolchains requires that implementation first.

8 Conclusion

AEGIS introduces seven typed EA entities: Evidence, Obligation, Policy, Risk, RuntimeSignal, Exception, ArchitectureDecisionRecord. Seven typed relations

connect them to the ArchiMate Next metamodel. The IMVDOA lifecycle replaces TOGAF's ADM with a continuous loop. The regulatory crosswalk covers all 38 ISO/IEC 42001 controls and the key EU AI Act Articles for high-risk systems.

The Accountability Fabric [12] is the closest structural predecessor. AEGIS extends it to LLM-based agents, adds a runtime loop, and adds a regulatory crosswalk to the EU AI Act and ISO/IEC 42001.

Three expressiveness constructions show that AEGIS can represent all 38 ISO 42001 Annex A controls as typed obligations, detect three classes of obligation drift by logical inference, and integrate Policy Cards into an EA-level conformance graph that exposes inter-card conflicts. Longitudinal evaluation against production toolchains remains future work.

Data Availability

No datasets were generated or analysed in this study. All artefacts are defined within the paper itself.

References

1. AI, N.: Artificial intelligence risk management framework (ai rmf 1.0). URL: https://nvlpubs. nist. gov/nistpubs/ai/nist. ai pp. 100–1 (2023)
2. Anthropic: Model context protocol (MCP). Open technical specification (2024), https://modelcontextprotocol.io
3. CycloneDX / ECMA International, Linux Foundation SPDX: AI Bill of Materials: CycloneDX ML-BOM v1.7 (ECMA-424) and SPDX 3.0 AI Profile. Technical Standards (2025), https://cyclonedx.org/capabilities/mlbom/
4. 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, L 2024/1689 (2024), https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L_202401689
5. Ford, N., Richards, M.: Architecture as Code. O'Reilly Media (2025), forthcoming
6. 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: Towards a Knowledge-Aware AI: SEMANTiCS 2022—Proceedings of the 18th International Conference on Semantic Systems, 13–15 September 2022, Vienna, Austria. pp. 51–65. SAGE Publications Pvt. Ltd 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP (2022)
7. Golpayegani, D., Pandit, H.J., Lewis, D.: Comparison and analysis of 3 key ai documents: Eu's proposed ai act, assessment list for trustworthy ai (altai), and iso/iec 42001 ai management system. In: Irish Conference on Artificial Intelligence and Cognitive Science. pp. 189–200. Springer (2022)
8. Hernandez, J., Golpayegani, D., Lewis, D.: An open knowledge graph-based approach for mapping concepts and requirements between the eu ai act and international standards. AI and Ethics **5**(5), 4463–4474 (2025)
9. Keeling, M.: Love unrequited: The story of architecture, agile, and how architecture decision records brought them together. IEEE Software **39**(4), 90–93 (2022)

10. Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J.: Prov-o: The prov ontology (2013)
11. Lewis, D., Lasek-Markey, M., Golpayegani, D., Pandit, H.J.: Mapping the regulatory learning space for the eu ai act (2025), https://arxiv.org/abs/2503.05787
12. Markovic, M., Naja, I., Edwards, P., Pang, W.: The accountability fabric: A suite of semantic tools for managing ai system accountability and audit. In: ISWC (Posters/Demos/Industry) (2021)
13. Mavračić, J.: Policy cards: Machine-readable runtime governance for autonomous AI agents. arXiv:2510.24383 (Oct 2025), https://arxiv.org/abs/2510.24383
14. Neufeld, E.A., Bartocci, E., Ciabattoni, A., Governatori, G.: Enforcing ethical goals over reinforcement-learning policies. Ethics and Information Technology **24**(4), 43 (2022)
15. Ojewale, V., Suresh, H., Venkatasubramanian, S.: Audit trails for accountability in large language models (2026), https://arxiv.org/abs/2601.20727
16. Pandit, H.J., Esteves, B., Krog, G.P., Ryan, P., Golpayegani, D., Flake, J.: Data privacy vocabulary (dpv)–version 2. arXiv preprint arXiv:2404.13426 (2024)
17. Schnabl, C., Hugenroth, D., Marino, B., Beresford, A.R.: Attestable audits: Verifiable ai safety benchmarks using trusted execution environments (2025), https://arxiv.org/abs/2506.23706
18. The Open Group: TOGAF Standard, 10th Edition. The Open Group (2022), https://www.opengroup.org/togaf
19. The Open Group: ArchiMate 3.2 Specification. The Open Group (2023), https://www.opengroup.org/archimate-forum/archimate-overview
20. The Open Group: ArchiMate next specification snapshot 1. Tech. rep., The Open Group (2025), https://publications.opengroup.org/s250, working Draft