Automated AI Transparency: Dynamic Ontology Instantiation for MLOps

Fabian Kovac[1], Dimitrios Christodoulou[2], Tobias Dam[1] and Sebastian Neumaier[1,3]

[1]University of Applied Sciences St. Pölten, Austria
[2]Digital for Planet, Switzerland
[3]Vienna University of Economics and Business, Austria

Abstract

While emerging regulations mandate strict transparency for AI systems, modern Machine Learning Operations (MLOps) pipelines remain primarily sub-symbolic, managing opaque neural artifacts, raw data, and numerical metrics. This presents a challenge for automated regulatory oversight. In this paper, we present an implementation of a Semantic MLOps Engine that establishes a neuro-symbolic lifecycle for AI development. By dynamically tracking sub-symbolic ML processes and instantiating them into a formal Knowledge Graph based on the AIDOC-AP ontology, we bridge the gap between model execution and symbolic governance. Furthermore, we demonstrate a Completeness Dashboard that evaluates the instantiated graph through ontology-mediated SPARQL competency queries to compute documentation completeness. Using a real-world energy forecasting scenario, we illustrate how our approach automates the acquisition and validation of documentation evidence, and makes the residual human-provided items explicit, without disrupting the developer workflow.

Keywords

Semantic MLOps, AI Transparency, Compliance, AI System

1. Introduction

With the EU AI Act [1], transparency has become a legally binding property of AI systems. Article 11 requires providers of high-risk AI systems to draw up technical documentation *before* a system is placed on the market and to keep it up to date thereafter; Annex IV enumerates the minimum contents of this documentation, ranging from the general logic and architecture of the system, over the provenance and preparation of training, validation and testing data, to performance metrics, risk management and post-market monitoring. Documentation under the AI Act is thus not a one-off deliverable, but a continuous obligation that spans the entire lifecycle of a system.

This obligation stands in contrast to how AI systems are actually built and operated today. Modern development is organised around Machine Learning Operations (MLOps) toolchains (experiment trackers, model registries, automated training and deployment pipelines) whose artifacts are inherently sub-symbolic [1]: model weights, serialised datasets, numerical metrics and hyperparameters. The notions regulators care about, such as intended purpose, data provenance and human oversight, are not first-class citizens of an MLOps platform. In practice, technical documentation is therefore written manually and retrospectively, e.g., in the form of model cards [2] and datasheets [3], and inevitably drifts away from the system it is supposed to describe. What is missing is a machine-interpretable, symbolic representation of the ML lifecycle that can be queried, validated and kept in sync with the running system.

Third Workshop on Knowledge Graphs and Neurosymbolic AI (KG-NeSy 2026), in ISWC 2026 Workshops Joint Proceedings, October 25–26, 2026, Bari, Italy

fabian.kovac@ustp.at (F. Kovac); dimitrios.christodoulou@digital4planet.org (D. Christodoulou); tobias.dam@ustp.at (T. Dam); sebastian.neumaier@ustp.at (S. Neumaier)
0000-0002-1611-2598 (F. Kovac); 0000-0002-2463-5831 (T. Dam); 0000-0002-9804-4882 (S. Neumaier)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1]We use *sub-symbolic* in a systems-level sense: the artifacts of the ML lifecycle (model weights, serialised datasets, numerical metrics, hyperparameters) carry no machine-interpretable semantics. The term does not imply that the symbolic layer accesses or interprets the internal representations of the models.

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-05

CEUR-WS.org/Vol-4313/kgnesy-paper12.pdf

We argue that closing this gap is a neuro-symbolic integration problem at the systems level [4, 5]: rather than injecting symbols into model internals, we couple the sub-symbolic lifecycle with a symbolic layer that reasons over its operational footprint. The sub-symbolic layer, i.e., training, evaluating and serving machine-learned models, remains the executing substrate of the AI system, while its footprint is continuously lifted into a knowledge graph whose nodes and edges are typed by a formal ontology of AI documentation. Over this auxiliary symbolic structure, established Semantic Web techniques become applicable to the certification and auditing of the AI system: ontology-mediated SPARQL queries, rewritten by Ontop with respect to the AIDOC-AP axioms, answer competency questions derived from legal requirements, and this validation determines whether the documentation of a system is complete. Crucially, the two layers form a closed loop rather than a one-way reporting channel: every sub-symbolic lifecycle event (a training run, a model registration, a dataset update) is reflected as symbolic statements; the symbolic layer identifies which evidence is still missing; and this gap report steers the sub-symbolic process (what is logged, evaluated and changed next), which is in turn re-instantiated into the graph. Two scoping remarks are in order. First, the integration operates at the systems level: the symbolic layer represents and queries the operational footprint of the ML lifecycle, not the internal representations or learning dynamics of the models themselves. Second, the loop is deliberately closed through a human in the loop: the gap report informs an engineer who reconfigures the pipeline, rather than the symbolic layer acting on the sub-symbolic component directly. In terms of the boxology of hybrid learning and reasoning systems [6], our architecture is thus a compositional design pattern in which the observable behaviour of a neural component feeds a symbolic component whose output, mediated by a human, governs the further development of that component.

In this paper, we present an implementation of this neuro-symbolic lifecycle, developed in the Horizon Europe project CERTAIN.[2] A companion position paper [7] outlines the methodological steps envisaged for the CERTAIN framework: semantic MLOps for structured lifecycle management, ontology-driven lineage tracking, and regulatory operations (RegOps) workflows. The present paper realises the first two of these as running software and evaluates them on a production pilot. Our contributions are:

- a *Semantic MLOps Engine* that tracks the sub-symbolic ML lifecycle of an MLflow-based [8] pipeline and dynamically instantiates its artifacts into a knowledge graph conforming to the AIDOC-AP application profile, without requiring a separate documentation workflow;
- a *Completeness Dashboard* that evaluates the instantiated knowledge graph against the formalised documentation requirements of AI Act Annex IV and reports documentation completeness as a coverage score with actionable gaps; and
- a demonstration of the end-to-end approach on a real-world energy demand forecasting system from one of the CERTAIN pilots.

The AIDOC-AP ontology itself, including its engineering methodology, alignments to reference ontologies and evaluation, is presented in a separate resource paper [9]; here, it plays the role of the documentation schema that the engine writes and the dashboard reads.

The remainder of this paper is structured as follows: Section 2 gives an overview of the system architecture and introduces AIDOC-AP as its documentation artifact. Section 3 describes the dynamic ontology instantiation via the Semantic MLOps Engine, including the energy forecasting running example, and Section 4 presents the coverage report and dashboard. Section 5 concludes with an outlook on the integration with further CERTAIN pilots.

2. System Architecture

Figure 1 illustrates the proposed architecture, which realises the neuro-symbolic lifecycle outlined above by extending a conventional MLOps workflow with a symbolic layer for automated transparency reporting.

[2] https://certain-project.eu/

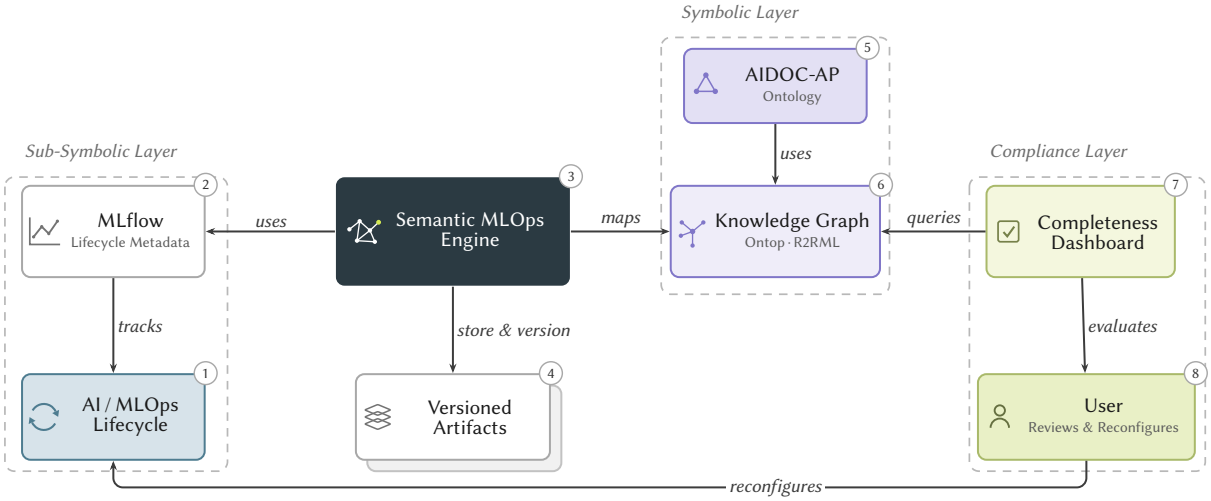


Figure 1: System architecture for automated AI transparency. (1) The AI/MLOps lifecycle comprises data preparation, model training, and deployment stages. (2) MLflow [8] instruments this lifecycle and captures the metadata produced at each stage. (3) The Semantic MLOps Engine automatically collects this metadata across all lifecycle stages and (4) stores the resulting artifacts under version control. (5) The AIDOC-AP ontology [9] provides the conceptual schema for (6) R2RML [10] mappings, which expose the structured metadata as a virtual knowledge graph via Ontop [11]. (7) The Completeness Dashboard queries the knowledge graph to assess compliance readiness, and (8) a human in the loop reviews the results and reconfigures the pipeline to achieve compliance.

The sub-symbolic substrate is an MLflow-based [8] pipeline that tracks experiments, models and metrics across the AI lifecycle. At the core of the architecture, the Semantic MLOps Engine intercepts the metadata that MLflow records at every lifecycle stage and persists it, together with the associated artifacts, in a versioned and queryable form.

The lifting into the symbolic layer follows an ontology-based data access (OBDA) approach: rather than materialising the metadata into a separate triple store, declarative R2RML [10] mappings [3] expose it as a virtual knowledge graph through Ontop [11], so that the symbolic view remains synchronised with the running system at all times and documentation drift is prevented by construction. The Completeness Dashboard queries this knowledge graph and quantifies to what extent the required documentation evidence is present with respect to the formalised requirements of Article 11 and Annex IV of the EU AI Act, reporting coverage together with the concrete evidence that is still missing.

This gap report closes the loop: guided by the dashboard, the user reconfigures the learning pipeline, and the resulting lifecycle events are in turn re-instantiated into the graph. The semantic backbone of the architecture is the AIDOC-AP application profile [9], which types all collected metadata and is introduced in Section 2.1.

2.1. AIDOC-AP as Documentation Artifact

AIDOC-AP[4] is an application profile for the technical documentation of AI systems, grounded in the documentation obligation of Article 11 and Annex IV of the EU AI Act. It is implemented as an OWL ontology that reuses and aligns with established vocabularies, including AIRO [12] and VAIR [13] for AI risk concepts, PROV-O [14] for provenance, ML Schema [15] for machine learning experiments, DCAT [16] and DQV [17] for datasets and data quality, and DPV [18] for privacy and AI Act concepts. The engineering methodology behind the profile, its alignments and its evaluation are described in the accompanying resource paper [9]; in the following, we focus on the role the profile plays within

[3] https://github.com/CERTAIN-Project/Semantic_MLOps_engine/blob/main/ontop/input/aidoc-ap_r2rml.ttl
[4] https://w3id.org/aidoc-ap

our architecture: it is the schema that the Semantic MLOps Engine writes and that the Completeness Dashboard reads.

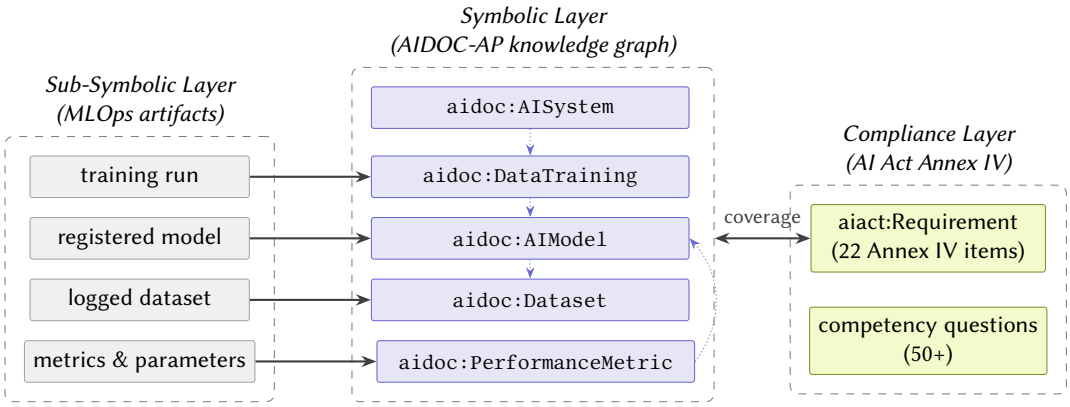


Figure 2: AIDOC-AP as the symbolic target schema: sub-symbolic MLOps artifacts are dynamically instantiated as typed knowledge graph resources (Section 3), which are then evaluated against the formalised Annex IV requirements and their competency questions (Section 4).

For this data flow, three groups of terms matter. First, the *core entities*: an `aidoc:AISystem` acts as the documentation anchor and is linked to its software and hardware components, interfaces, capabilities and intended purpose, as well as to the `aidoc:AIModels` it employs and the `aidoc:Datasets` these were built from. Second, the *lifecycle activities*: the profile models the data pipeline (acquisition, wrangling, cleaning and labeling procedures, and the training, validation and testing of data), the ML pipeline (model engineering, evaluation with associated `aidoc:PerformanceMetrics`, packaging and versioning) and the deployment and monitoring phases. Third, the *documentation artifacts and agents* required by Annex IV: datasheets, logs, change records, transparency measures and human oversight mechanisms, connected to the responsible agents such as provider, deployer and auditor. Each of these classes deliberately corresponds to something an MLOps platform already manages: a registered model becomes an `aidoc:AIModel`, a logged dataset an `aidoc:Dataset`, a training run a lifecycle activity, and a logged evaluation result an `aidoc:PerformanceMetric` (Figure 2). Instantiating the profile is thus largely a typing exercise over existing operational metadata rather than an additional authoring task; this observation is what the Semantic MLOps Engine in Section 3 operationalises.

Finally, AIDOC-AP also formalises the requirements it is meant to satisfy: a companion module represents the documentation items of Annex IV as 22 `aiact:Requirement` individuals, each traceably linked to the underlying legal text and to more than 50 competency questions that express what a compliant documentation graph must be able to answer. These requirements serve as the normative yardstick of the framework: the coverage report in Section 4 evaluates the instantiated knowledge graph against them. From the perspective of the overall system, AIDOC-AP is therefore less a vocabulary than a contract: the engine guarantees that documentation is produced in this shape, and the dashboard measures how much of the contract is already fulfilled.

3. Dynamic Ontology Instantiation via Semantic MLOps

The Semantic MLOps Engine extends a conventional MLOps infrastructure with an automated and continuous acquisition of sub-symbolic lifecycle artifacts. It captures the operational footprint generated throughout the machine-learning lifecycle, including data collection and annotation, experimentation, training, evaluation, model registration, deployment, and post-deployment adaptation.

3.1. Automated Lifecycle Tracking

Conventional MLOps platforms support experiment tracking, model management, and deployment, but they generally preserve only the information generated by the platform or explicitly logged by

developers. As a result, the lifecycle metadata required to document and assess an ML system (dataset characteristics, data-processing steps, evaluation procedures, runtime conditions, model lineage, deployment history, sustainability indicators, and governance decisions) remains distributed across heterogeneous tools, files and repositories. Especially for high-risk AI systems, organizations must identify the required evidence, build pipelines to collect and store it, reconstruct relationships among the resulting records, and largely manually analyze the system state before producing documentation or compliance reports, a costly and error-prone process.

The Semantic MLOps Engine [5] extends, rather than replaces, an existing MLOps infrastructure and automates this metadata collection. The core innovation is that the acquisition process is ontology-driven: the target ontologies determine not only how collected metadata is normalized and semantically represented, but also which lifecycle metadata must be captured, computed, or explicitly supplied. These specifications are operationalized through tracking and metadata-management functions that extend conventional tools such as MLflow [8] and can be embedded directly into existing machine-learning pipelines, as illustrated in Listing 1. A FastAPI-based data-transfer service [19] then transforms the collected metadata into a normalized representation in relational storage, while the semantic mappings are developed manually offline (Section 3.2).

During ordinary machine-learning cycles, the engine captures metadata about datasets, annotation and preprocessing activities, experiments, training runs, parameters, performance measurements, model registration, deployments, runtime events, source-code versions, execution environments, resource consumption, and responsible actors. Specialized components cover what existing tools do not provide directly, such as dataset profiling through WhyLogs [20] (`log_whylogs_profile` in Listing 1) and resource and sustainability measurements through CodeCarbon [21] (`start_tracker`/`stop_tracker`). Information that cannot be automatically retrieved, such as intended use, contextual limitations, risk-management decisions, or organizational approvals, is obtained through explicit developer or stakeholder interactions via dedicated metadata-management functions of the engine and the data-transfer service's API, which validate the supplied items against the expected AIDOC-AP terms and store them alongside the automatically captured metadata. The engine therefore integrates automatically captured, computationally derived, and human-provided lifecycle metadata within a common processing framework.

Synchronisation is event-driven: completion of a tracked activity (e.g. a training run or deployment) triggers the data-transfer service, which retrieves the MLflow run and its artifacts and normalizes them into the relational schema; the stored records are exposed as a virtual knowledge graph typed by the AIDOC-AP [9] ontology via the mappings described in Section 3.2, so that new records are visible on the next query without any re-materialisation. Operational artifacts such as datasets, training executions, AI models, evaluation results, deployments, and provenance records become ontology instances with stable identifiers, minted from primary keys via the shared IRI templates of the R2RML mappings (cf. Listing 2), and explicit semantic relations, so that heterogeneous records from different tools are accessible through the same conceptual vocabulary, independently of their original schemas, file formats, or storage systems.

The resulting knowledge graph is not a static semantic export: every completed run can introduce new entities, update lifecycle states, and establish additional relations. A metric is represented not merely as a numeric value, but in relation to the model it evaluates, the run that produced that model, and the datasets used during training or testing. This semantic context enables SPARQL-based lifecycle queries and ontology-based compliance assessment, and reduces repeated manual status reconstruction, since the graph reflects the evidence already generated by the operational system.

3.2. R2RML Mappings and the Virtual Knowledge Graph

The connection between the ontology and the knowledge graph is established through declarative R2RML [10] mappings. R2RML, a W3C Recommendation for expressing customised mappings from

[5] https://github.com/CERTAIN-Project/Semantic_MLOps_engine

```python
from semantic_mlops import tracker

tracker.set_experiment(experiment_name="Energy Prediction", tags=experiment_tags)
with tracker.start_run(run_name="rf_regressor", tags=run_tags) as run:
    # CodeCarbon and WhyLogs
    tracker.log_git_metadata()
    tracker.start_tracker(output_file_name="emissions_data")
    tracker.save_dataset_manifest(data)
    tracker.log_data_techniques(techniques_dict)
    tracker.log_dataset(data)
    tracker.log_whylogs_profile(data)
    tracker.stop_tracker()

    # per search iteration:
    tracker.log_search_space(search_space_dict)
    model.fit(X_train, y_train)
    tracker.log_params(params_dict)
    tracker.log_metrics(metrics_dict, step=step)
    tracker.log_resources(resources_dict, step=step)
    tracker.log_model_signature(model)
```

Listing 1: Automated lifecycle tracking with the Semantic MLOps Engine: the tracking calls are embedded in the pipeline's training script and executed on the target system (in this case, the energy pilot) on every training run.

relational databases to RDF, organises a mapping as a set of *triples maps*: each triples map pairs a logical table (a base table, view or SQL query over the normalised metadata schema) with a *subject map* that mints stable IRIs from primary keys via IRI templates and assigns the corresponding AIDOC-AP class, and with *predicate-object maps* that translate columns and foreign-key references, embedded in shared IRI templates, into ontology properties. The correspondences sketched in Section 2.1 are realised exactly here, as illustrated in Listing 2: a registered-model record yields an `aidoc:AIModel` individual, the evaluation rows attached to a run yield its `aidoc:PerformanceMetrics`, and the foreign keys between them yield the lifecycle relations that connect these individuals in the graph. The AIDOC-AP ontology therefore acts as the global schema of an OBDA setting: Ontop [11] interprets the mappings at query time and rewrites SPARQL queries over the AIDOC-AP vocabulary into SQL over the underlying records, so that no triples are materialised and every change in the relational layer is immediately visible in the symbolic layer. Adapting the documentation schema (e.g., when a new Annex IV item requires additional evidence) amounts to extending the ontology and the affected triples maps, leaving the acquisition pipeline untouched.

```
<#ModelTriplesMap> a rr:TriplesMap ;
  rr:logicalTable [ rr:tableName "model_architecture" ] ;
  rr:subjectMap [
    rr:template "https://w3id.org/aidoc-ap/models/{run_id}/{model_id}" ;
    rr:class aidoc:AIModel ; rr:class mls:Model ] ;
  rr:predicateObjectMap [
    rr:predicate aidoc:usesAIMethod ;
    rr:objectMap [
      rr:template "https://w3id.org/aidoc-ap/methods/{architecture_name}" ;
      rr:termType rr:IRI ] ] ;
  # link to the producing training run via foreign key
  rr:predicateObjectMap [
    rr:predicate prov:wasGeneratedBy ;
    rr:objectMap [
      rr:template "https://w3id.org/aidoc-ap/runs/{run_id}" ;
      rr:termType rr:IRI ] ] .
```

Listing 2: Condensed excerpt of the R2RML mappings: model records are lifted into `aidoc:AIModel` individuals; the foreign-key column `run_id` appears in a shared IRI template, so the relational reference materialises as the linking triple (`prov:wasGeneratedBy`) that connects the model to its producing training run in the graph.

```
@prefix aidoc: <https://w3id.org/aidoc-ap#> .
@prefix airo:  <https://w3id.org/airo#> .
@prefix dpv:   <https://w3id.org/dpv#> .
@prefix dqv:   <http://www.w3.org/ns/dqv#> .

:Training_Activity a aidoc:DataTraining ;
   aidoc:hasFrequency freq:daily ;
   aidoc:usesTrainingData
      :Combined_Energy_and_Weather_Data .

:Combine_Datasets_Activity a aidoc:DataProcessing ;
   aidoc:hasSourceDataset :Energy_Data, :Weather_Data ;
   aidoc:producesDataset
      :Combined_Energy_and_Weather_Data .

:Energy_Data a aidoc:Dataset ;
   dcterms:description """readings of metering points,
      where each reading has a corresponding
      quality flag""" ;
   dpv:hasDataSource :EEGFaktura ;
   dqv:hasQualityMeasurement
      :Accuracy, :Completeness, :Consistency,
      :Timeliness .

:Encom_AI_Model a aidoc:AIModel ;
   aidoc:usesAIMethod :XGBoost ;
   airo:hasComponent :Hyperparameters .

:XGBoost a aidoc:AIMethod ;
   rdfs:label "Extreme Gradient Boosting" ;
   airo:producesOutput :Predictions .
```

Listing 3: Excerpt of the instantiated Encom knowledge graph: a daily training event is lifted into an `aidoc:DataTraining` activity whose training data is traceable through the combining `aidoc:DataProcessing` step to the source datasets, their external data sources and attached quality measurements, and connected to the `aidoc:AIModel` it produces.

3.3. Running Example: Encom Energy Forecasting System

We demonstrate the dynamic instantiation on *Encom*, an energy demand and generation forecasting system operated within the CERTAIN energy pilot. Encom provides short-term forecasts of energy consumption and generation for renewable energy communities [22] in Austria, both for single metering points and for semantic groups of metering points [23], and additionally predicts optimal schedules of participation factors. Operationally, the system is a containerised multi-service deployment: Apache Airflow [24] orchestrates the data and training pipelines, MLflow [8] tracks experiments and model versions, a TimescaleDB instance stores the time-series data, and a MinIO object store holds datasets, predictions and log backups. The forecasting models are gradient-boosted tree ensembles (XGBoost) [25] whose hyperparameters are re-optimised on a monthly basis via Bayesian optimisation [26], while the models themselves are retrained daily on the most recent data.

From a documentation perspective, two artifact groups are central. On the data side, the *Energy Dataset* contains metering-point readings ingested from an external energy-community billing service, where each reading carries a quality flag assigned by the grid operator; only quality-confirmed data points enter the training process, a cleaning procedure the profile captures as an `aidoc:DataCleaningProcedure`. The energy data is joined with weather observations from GeoSphere Austria [6] and day-ahead exchange prices from the EPEX spot market retrieved from the ENTSO-E Transparency Platform [7] into a combined training dataset, and the associated data-quality dimensions (accuracy, completeness, consistency, timeliness, among others) are attached as `dqv:QualityMeasurements` [17]. On the model side, the *Forecasting Model* is represented as an `aidoc:AIModel` that uses XGBoost as its `aidoc:AIMethod` and produces the prediction dataset served to the energy communities; its versioning is handled by MLflow

and reflected as an `aidoc:ModelVersioning` activity within the development lifecycle stage.

Listing 3 shows the symbolic footprint of a single daily training event as it materialises in the knowledge graph. None of these triples are authored manually: the tracking calls of Listing 1 run as part of the Airflow pipeline, the Semantic MLOps Engine captures the resulting MLflow run together with the logged input datasets, and the R2RML mappings of Section 3.2 expose the normalised records as typed AIDOC-AP resources. The training run surfaces as an `aidoc:DataTraining` activity with a declared daily frequency; the datasets it consumes are linked back through an `aidoc:DataProcessing` step to their source datasets and ultimately to the external data sources, yielding an end-to-end provenance chain from the trained model to the originating metering-point readings, as required by the data-related items of Annex IV.

Beyond the training pipeline, the instantiated graph anchors these artifacts in the wider system documentation introduced in Section 2.1: the `aidoc:AISystem` individual links the model and its implementation to thirteen software components across four servers, three formalised intended purposes, lifecycle stages from development to operation, and change records such as dependency updates. Risk-related individuals (e.g., data poisoning as a risk source and user training as a mitigating control) are included to exercise the corresponding parts of the profile; as they do not stem from the operational pipeline, they are marked as fictional via a dedicated annotation property. The resulting graph comprises roughly 70 individuals and forms the input to the coverage evaluation in Section 4.

4. AIDOC-AP Coverage Report

Compliance assessment is operationalised as a graph-validation task: rather than treating Annex IV conformity as a narrative checklist, we execute a suite of 50 SPARQL queries, each aligned with one competency question from Section 2.1, and test whether the required statements are present in the instantiated AIDOC-AP graph. The result is a completeness score that distinguishes available from missing documentation evidence and turns compliance assessment into a repeatable, machine-supported validation step.

The Completeness Dashboard makes this workflow usable in engineering practice: it evaluates the instantiated graph against the formalised Annex IV requirement model in AIDOC-AP, executing the competency-query set server-side via Comunica [27], and is implemented as a SvelteKit application with TypeScript and a Flowbite/Tailwind interface.[8] It provides a traffic-light summary of passed, failed and partially fulfilled checks, an overall coverage percentage, and query-level inspection with filtering and detailed result feedback, thereby serving not only as a reporting interface but as a remediation instrument that translates missing evidence into concrete follow-up actions. The dashboard itself invokes no ontology reasoner: it relies on SPARQL execution over the instantiated triples, with ontology-level inference confined to Ontop's query rewriting over the AIDOC-AP axioms.

Listing 4 illustrates how an individual competency question is operationalised at query level. The query retrieves security measures linked either directly to the documented AI system or via related risk-control assertions, together with optional labels, descriptions and affected risk concepts. If no matching bindings are returned, the corresponding requirement is flagged as insufficiently evidenced and contributes to the remaining coverage gap.

4.1. Dashboard Demonstration

We demonstrate the practical applicability of the Completeness Dashboard using the Encom example instantiation of the AIDOC-AP ontology, as detailed in Section 3.3, following the same workflow an engineer would use in practice: the SPARQL endpoint of the instantiated knowledge graph is configured for the dashboard, and the analysis is started, triggering the execution of the competency-question query suite against the endpoint.

[6] https://data.hub.geosphere.at/en/

[7] https://www.entsoe.eu/data/transparency-platform/

[8] https://svelte.dev/docs/kit/introduction, https://flowbite-svelte.com/

```sparql
PREFIX aidoc: <https://w3id.org/aidoc-ap#>
PREFIX airo: <https://w3id.org/airo#>
PREFIX vair: <https://w3id.org/vair#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?system ?systemLabel ?securityMeasure ?measureLabel ?measureDescription ?mitigatesRisk
WHERE {
  ?system a aidoc:AISystem .
  OPTIONAL { ?system rdfs:label ?systemLabel . }
  {
    ?securityMeasure a vair:SecurityMeasure .
    OPTIONAL { ?securityMeasure rdfs:label ?measureLabel . }
    OPTIONAL { ?securityMeasure dcterms:description ?measureDescription . }
    OPTIONAL { ?securityMeasure airo:modifiesRiskConcept ?mitigatesRisk . }
  } UNION {
    ?system airo:hasRiskControl ?securityMeasure .
    OPTIONAL { ?securityMeasure rdfs:label ?measureLabel . }
    OPTIONAL { ?securityMeasure dcterms:description ?measureDescription . }
  }
}
```

Listing 4: Example competency query (CQ15.1) used in the AIDOC-AP Coverage Report: the SPARQL pattern checks whether the documentation graph contains evidence for Requirement 15 on cybersecurity measures.

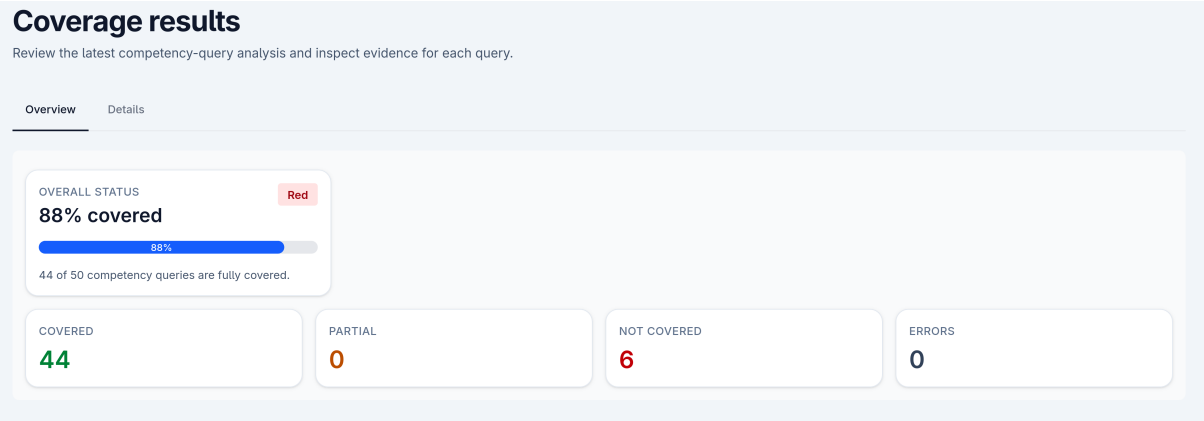


Figure 3: The *Overview* tab of the Completeness Dashboard shows the overall coverage percentage and the number of competency questions that are covered, partially covered, not covered, or returned errors.

After execution, the *Overview* view provides the aggregate compliance status, including the overall coverage percentage and the number of competency questions that are covered, partially covered, not covered, or returned errors, as shown in Figure 3. The *Detail* view exposes the per-question evidence trail: for each competency question, it reports the status (covered, partial, not covered, or error), the evidence as the number of query result bindings, the associated Annex IV requirement, the SPARQL query file that was executed, and the returned result set. Figure 4 shows the detail view for a specific competency question that is not covered by the instantiated knowledge graph, allowing users to move directly from aggregate coverage values to concrete remediation actions. Each report reflects the state of the virtual graph for the latest run, in which model and dataset versions are first-class individuals (e.g., `aidoc:ModelVersioning`); a dedicated property stamping the report with a specific pipeline release is not yet implemented and is part of the longitudinal tracking outlined in Section 5.

For the Encom instantiation, the dashboard reports 88% coverage: 44 of the 50 competency queries are answered by the captured evidence, while six remain uncovered (Figure 4), concerning training datasheets, labelling procedures, human oversight measures, technical measures facilitating output interpretation, validation and testing data characteristics, and adopted harmonised standards. Of the covered questions, the risk-related ones (including CQ15.1, Listing 4) are answered by the fictional

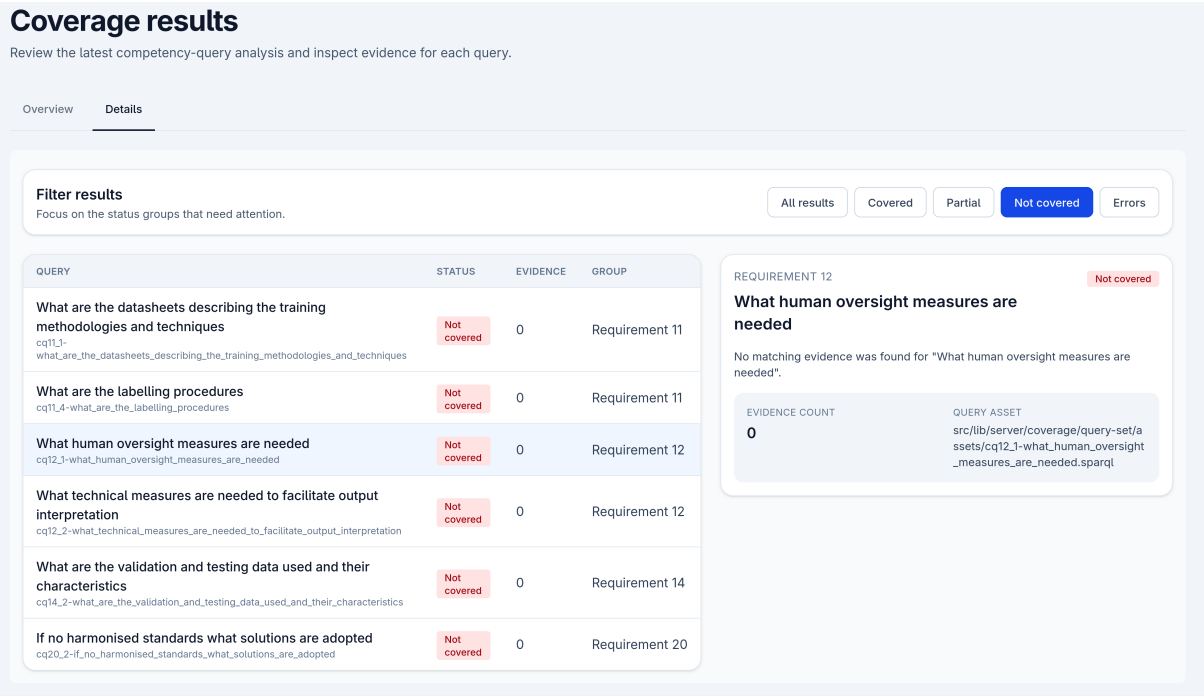


Figure 4: The *Details* tab of the Completeness Dashboard shows the competency questions that are not covered by the instantiated knowledge graph among others. The user can inspect the evidence trail for each question and identify the missing symbolic links required by the application profile.

risk individuals of Section 3.3 and thus demonstrate the capability of the profile and query suite rather than operationally sourced risk documentation. Notably, these gaps correspond almost entirely to the *human-provided* acquisition mode identified in Section 3, rather than to artifacts the engine failed to track: they are documentation items that no pipeline can generate on its own. The gap report thus does more than quantify incompleteness: it separates the evidence that automation already delivers from the residual items stakeholders still have to supply, and it points the engineer to the tracking and metadata-management functions through which this evidence enters the graph in the next synchronisation cycle.

5. Conclusion and Future Work

This paper presented an end-to-end implementation of a neuro-symbolic lifecycle for automated AI transparency, developed in the Horizon Europe project CERTAIN. The Semantic MLOps Engine continuously lifts the footprint of an MLflow-based pipeline (runs, datasets, models, metrics and their provenance) into a virtual knowledge graph typed by the AIDOC-AP application profile, and the Completeness Dashboard evaluates this graph against the formalised documentation requirements of AI Act Annex IV. Applied to an energy forecasting pilot, the approach yields technical documentation whose completeness is computed rather than asserted: 44 of 50 competency queries are answered by evidence captured during ordinary pipeline operation, and the remaining gaps are pinpointed as concrete documentation tasks. The demonstration supports the central design hypothesis of this work: much of the evidence Annex IV asks for already exists as operational metadata and can be documented automatically; where it cannot, the symbolic layer makes the residual human effort explicit and actionable.

Limitations. Our evaluation demonstrates functionality on a single pilot; runtime and storage overhead, OBDA scalability, and the effort saved over manually authored documentation remain

unquantified. Coverage measures whether required evidence is present, not whether it is correct or legally sufficient; the dashboard supports engineers and auditors rather than substituting conformity assessment. Finally, the demonstrated neuro-symbolic coupling is a system-level integration whose loop is closed by a human; reasoning about model internals or the learning process is out of scope.

Future work proceeds along three lines. First, we will roll the engine and dashboard out to further CERTAIN pilots in distinct AI Act high-risk domains, including health and biometrics: the R2RML mappings target the engine's normalized schema and transfer directly, so the open question is whether the tracking functions and the schema capture the domain-specific evidence these pilots require. Second, we plan to track coverage longitudinally, so that every release of an AI system carries a documentation score and regressions in transparency become as visible as regressions in accuracy. Third, we aim to deepen the neuro-symbolic coupling itself: LLM-assisted enrichment of unstructured artifacts (e.g., free-text descriptions and logs), validated against the symbolic layer before ingestion, and dashboard gaps translated into pre-configured tracking prompts would further shorten the path from a detected gap to its remediation. Because the requirements are themselves modelled as ontology individuals, adaptations of Annex IV through delegated acts can be accommodated by evolving the requirements module and mappings, without changes to the engine, a property we consider essential for keeping automated transparency aligned with a moving regulatory target.

Declaration on Generative AI

During the preparation of this work, the authors used GitHub Copilot, Claude and ChatGPT in order to: draft content, paraphrase and reword, improve writing style, and check grammar and spelling. Claude and GitHub Copilot were further used to support the development of source code. After using these tools and services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] European Parliament and Council, Regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 on artificial intelligence (ai act), https://eur-lex.europa.eu/eli/reg/2024/1689/oj, 2024. Official Journal of the European Union L 206.

[2] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, T. Gebru, Model cards for model reporting, in: Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019, ACM, 2019, pp. 220–229. doi:10.1145/3287560.3287596.

[3] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. III, K. Crawford, Datasheets for datasets, Communications of the ACM 64 (2021) 86–92. doi:10.1145/3458723.

[4] P. Hitzler, A. Eberhart, M. Ebrahimi, M. K. Sarker, L. Zhou, Neuro-symbolic approaches in artificial intelligence, National Science Review 9 (2022) nwac035. doi:10.1093/nsr/nwac035.

[5] H. A. Kautz, The third AI summer: AAAI Robert S. Engelmore Memorial Lecture, AI Magazine 43 (2022) 105–125. doi:10.1002/aaai.12036.

[6] F. van Harmelen, A. ten Teije, A boxology of design patterns for hybrid learning and reasoning systems, Journal of Web Engineering 18 (2019) 97–124. doi:10.13052/jwe1540-9589.18133.

[7] F. Kovac, S. Neumaier, T. Pahi, T. Priebe, R. Rodrigues, D. Christodoulou, M. Cordy, S. Kubler, A. Kordia, G. Pitsiladis, J. Soldatos, P. Zervoudakis, Towards a framework for supporting the ethical and regulatory certification of AI systems, in: Proceedings of the Workshop on AI Certification, Fairness and Regulations, co-located with the Austrian Symposium on AI, Robotics and Vision (AIRoV 2025), 2025. arXiv:2510.00084.

[8] M. A. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, F. Xie, C. Zumar, Accelerating the Machine Learning Lifecycle

with MLflow, IEEE Data Eng. Bull. 41 (2018) 39–45. URL: https://api.semanticscholar.org/CorpusID:83459546.

[9] S. Neumaier, T. Dam, F. Kovac, AIDOC-AP: An application profile for technical documentation of AI systems, Semantic Web Journal (2026). Under review. Resource available at https://w3id.org/aidoc-ap/.

[10] S. Das, S. Sundara, R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C Recommendation, World Wide Web Consortium (W3C), 2012. URL: https://www.w3.org/TR/2012/REC-r2rml-20120927/, latest version: https://www.w3.org/TR/r2rml/.

[11] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Ontop: Answering SPARQL queries over relational databases, Semantic Web 8 (2017) 471–487. doi:10.3233/SW-160217.

[12] D. Golpayegani, H. J. Pandit, D. Lewis, AIRO: an ontology for representing AI risks based on the proposed EU AI act and ISO risk management standards, in: A. Dimou, S. Neumaier, T. Pellegrini, S. Vahdati (Eds.), Towards a Knowledge-Aware AI - SEMANTiCS 2022 - Proceedings of the 18th International Conference on Semantic Systems, 13-15 September 2022, Vienna, Austria, volume 55 of *Studies on the Semantic Web*, IOS Press, 2022, pp. 51–65. doi:10.3233/SSW220008.

[13] D. Golpayegani, H. J. Pandit, D. Lewis, To be high-risk, or not to be - semantic specifications and implications of the AI act's high-risk AI applications and harmonised standards, in: Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency, FAccT 2023, Chicago, IL, USA, June 12-15, 2023, ACM, 2023, pp. 905–915. doi:10.1145/3593013.3594050.

[14] T. Lebo, S. Sahoo, D. McGuinness, PROV-O: The PROV Ontology, W3C Recommendation, World Wide Web Consortium (W3C), 2013. URL: http://www.w3.org/TR/2013/REC-prov-o-20130430/, latest version: http://www.w3.org/TR/prov-o/.

[15] D. Esteves, A. Ławrynowicz, P. Panov, L. Soldatova, T. Soru, J. Vanschoren, ML Schema Core Specification, Technical Report, W3C Machine Learning Schema Community Group, 2016. URL: http://www.w3.org/2016/10/mls/.

[16] R. Albertoni, D. Browning, S. J. D. Cox, A. G. Beltran, A. Perego, P. Winstanley, Data Catalog Vocabulary (DCAT) – Version 3, W3C Recommendation, World Wide Web Consortium (W3C), 2024. URL: https://www.w3.org/TR/2024/REC-vocab-dcat-3-20240822/, latest version: https://www.w3.org/TR/vocab-dcat-3/.

[17] R. Albertoni, A. Isaac, Introducing the data quality vocabulary (DQV), Semantic Web 12 (2020) 81–97. doi:10.3233/SW-200382.

[18] H. J. Pandit, B. Esteves, G. P. Krog, P. Ryan, D. Golpayegani, J. Flake, Data privacy vocabulary (DPV) – version 2.0, in: The Semantic Web – ISWC 2024 – 23rd International Semantic Web Conference, Baltimore, MD, USA, November 11-15, 2024, Proceedings, Part III, volume 15233 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 171–193. doi:10.1007/978-3-031-77847-6_10.

[19] S. Ramírez, FastAPI, 2026. URL: https://github.com/fastapi/fastapi, accessed: 2026-07-17.

[20] WhyLabs, whylogs: The open standard for data logging, 2024. URL: https://github.com/whylabs/whylogs.

[21] B. Courty, V. Schmidt, S. Luccioni, Goyal-Kamal, MarionCoutarel, B. Feld, J. Lecourt, LiamConnell, A. Saboni, Inimaz, supatomic, M. Léval, L. Blanche, A. Cruveiller, ouminasara, F. Zhao, A. Joshi, A. Bogroff, H. de Lavoreille, N. Laskaris, E. Abati, D. Blank, Z. Wang, A. Catovic, M. Alencon, M. Stęchły, C. Bauer, L. O. N. de Araújo, JPW, MinervaBooks, mlco2/codecarbon: v2.4.1, 2024. doi:10.5281/zenodo.11171501.

[22] European Parliament, Council of the European Union, Directive (eu) 2018/2001 of the european parliament and of the council of 11 december 2018 on the promotion of the use of energy from renewable sources (recast), 2018. URL: https://eur-lex.europa.eu/eli/dir/2018/2001/oj/eng.

[23] I. Enne, M. Lackner, F. Kovac, T. Priebe, Scalable 72-hour generation and demand forecasting for renewable energy communities, in: 2026 11th International Conference on Smart and Sustainable Technologies (SpliTech), IEEE, 2026. In press.

[24] Apache Software Foundation, Apache airflow, 2026. URL: https://airflow.apache.org/, accessed: 2026-07-17.

[25] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 785–794. doi:10.1145/2939672.2939785.

[26] J. Snoek, H. Larochelle, R. Adams, Practical bayesian optimization of machine learning algorithms, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 25, Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

[27] R. Taelman, J. Van Herwegen, M. Vander Sande, R. Verborgh, Comunica: a modular sparql query engine for the web, in: Proceedings of the 17th International Semantic Web Conference, 2018. URL: https://comunica.github.io/Article-ISWC2018-Resource/.