A Microservice-based Architecture for Reproducible AI Pipelines

Moysis Symeonides
University of Cyprus
Nicosia, Cyprus
msymeo03@ucy.ac.cy

Joanna Georgiou
University of Cyprus
Nicosia, Cyprus
jgeorg02@ucy.ac.cy

George Pallis
University of Cyprus
Nicosia, Cyprus
pallis@ucy.ac.cy

Marios D. Dikaiakos
University of Cyprus
Nicosia, Cyprus
mdd@ucy.ac.cy

Dimitris Bouras
UBITECH
Athens, Greece
dbouras@ubitech.eu

Kostas Perakis
UBITECH
Athens, Greece
kperakis@ubitech.eu

André Grilo
UNINOVA-CTS and
School of Science and Technology
NOVA University of Lisbon
Caparica, Portugal
a.grilo@uninova.pt

Carlos Agostinho
UNINOVA-CTS and
Lab of Intelligent Systems (LASI) /
IDEA Institute
Caparica, Portugal / Funchal, Portugal
0000-0002-2884-776X

***Abstract*—As Artificial Intelligence (AI) systems move from prototypes to deployment, reliability and trustworthiness are increasingly limited by the data and AI pipeline lifecycle rather than by model training alone. Existing platforms offer strong support for individual lifecycle functions such as orchestration, experiment tracking, validation, or documentation, yet these capabilities are often adopted as separate tools, making provenance, observability, governance, and safe adaptation difficult to manage end-to-end. This paper presents the microservice-based reference architecture of the AI-DAPT project for reproducible AI pipelines whose core contribution is the unification of data- and model-centric operations within a single lifecycle-oriented platform. The architecture is organized around five iterative phases, namely data design, data sculpting, data generation, model delivery, and data/model optimization, while treating metadata, lineage, explainability, security, and human-oversight as native platform services rather than peripheral add-ons. The paper further contributes a trustworthiness-by-design architectural view that connects data preparation, valuation, synthetic data generation, orchestration, monitoring, adaptive retraining, and AI security through interoperable services. Finally, it presents the current realization of this architecture using widely adopted orchestration, storage, monitoring, and experimentation technologies, thereby offering a practical blueprint for building adaptable, observable, and governable AI pipelines in real-world settings.**

***Keywords*—MLOps, data-centric AI, trustworthy AI, microservices, pipeline orchestration, observability, explainable AI**

I. Introduction

Artificial intelligence (AI) is increasingly embedded in settings where decisions must remain reliable, traceable, and adaptable over time. In such environments, failures are often rooted less in the learning algorithm itself and more in the data and pipeline lifecycle. Dataset quality, provenance, semantic consistency, documentation, bias, distribution drift, and post-deployment monitoring all shape whether an AI system can be trusted, audited, and safely maintained. Yet in many deployments these activities are handled through fragmented tools and ad-hoc integrations, making it difficult to understand how data and model decisions interact and how systems evolve when requirements, environment, or data distributions change.

Existing Machine Learning (ML) platforms have substantially improved model-centric automation by supporting training, experiment tracking, pipeline execution, and deployment [1]–[3]. At the same time, complementary work has advanced important parts of the lifecycle, including dataset documentation [4], model transparency [5], data quality verification [6], and the analysis of technical debt in production ML systems [7]. However, these capabilities are still commonly adopted as separate platforms, or reporting frameworks, leaving practitioners to assemble heterogeneous stacks that can be difficult to maintain, audit, and evolve. Reusable support for data valuation, synthetic data generation, observability-driven adaptation, and governance-aware human intervention is rarely offered within a coherent end-to-end architectural view.

This gap becomes even more visible in settings where data- and model-centric processes must be jointly managed. Modern AI pipelines increasingly span heterogeneous data sources, iterative preparation and feature engineering steps, training and evaluation workflows, deployment environments, and continuous post-deployment adaptation. In addition, many industrial use cases require hybrid science-AI workflows, where data-driven components are combined with physics-based models, domain constraints, or rule-based systems to improve robustness and interpretability. Such workflows are increasingly recognized in MLOps and distributed AI architectures as a means of integrating learning-based and knowledge-driven components within a unified pipeline [8], [9]. These conditions call for an architectural approach that treats metadata, lineage, observability, reproducibility, interoperability, hybrid science-AI models, and security as first-class concerns throughout the lifecycle rather than as afterthoughts attached to isolated tools.

The AI-DAPT project [10] addresses this gap through a microservice-based architecture for reproducible and adaptive AI pipelines. Rather than introducing another isolated lifecycle

tool, this paper contributes an architectural framework that unifies capabilities usually fragmented across data engineering, MLOps, and trustworthy AI. Specifically, it makes three contributions: (i) it proposes a lifecycle-oriented reference architecture that unifies data- and model-centric operations across five iterative phases, from data design to post-deployment optimization; (ii) it introduces a trustworthiness-by-design view in which metadata, lineage, observability, explainability, security, and human oversight are embedded as native cross-lifecycle services, and (iii) it shows how these principles are realized through interoperable microservices built on widely used technologies. Thus, the contribution is not a new orchestration engine, tracker, or a validator artifact in isolation. Rather, it lies in the lifecycle unification captured by the architecture, while the technology stack shows a practical realization.

II. Related Work

Operational ML has been advanced by end-to-end platforms and MLOps toolchains that standardize pipeline execution, experiment tracking, artifact management, and deployment workflows. For instance, TensorFlow Extended (TFX) integrates data ingestion, validation, training, and serving into production-scale ML pipelines [1], while MLflow emphasizes portability across libraries through tracking, packaging, registry, and deployment support [3]. Moreover, Feature-store and AI platforms further extend MLOps data management, with Hopsworks providing a highly available feature store that supports feature reuse and consistent feature access from feature engineering to model training and inference across both offline and online workloads [2].

More broadly, survey work has consolidated common MLOps components and practices for production delivery and continuous operation [8]. Additionally, recent work has also proposed higher-level reference architectures for AI systems. For instance, Kreuzberger et al. [8] present an MLOps architecture that spans lifecycle stages, stakeholder roles, and infrastructure concerns for continuous ML delivery. Likewise, research on distributed AI has introduced architectures that combine heterogeneous components, coordination mechanisms, and data management across decentralized environments [9]. However, these works mainly describe broad architectural paradigms, while offering limited guidance on how such principles can be realized in an end-to-end platform that directly supports end users throughout their ML workflows.

A complementary group of works focuses on specific stages of the data pipeline. TensorFlow Data Validation (TFDV) supports scalable data profiling and anomaly detection [11], while more recent approaches enrich validation with domain-aware specifications and task-aware tests [12], [13]. In parallel, Datasheets for Datasets and Model Cards provide structured documentation for datasets and models to improve transparency, accountability, and responsible use [4], [5]. These contributions substantially improve individual lifecycle functions, but do not define how validation, documentation, provenance, and intervention are operationally coupled across AI pipelines. Recent efforts have also emphasized observability and operational intelligence. Observability-oriented approaches argue for end-to-end visibility for failure detection, diagnosis, and reaction [14], while AIOps research studies anomaly detection, event correlation, and failure management [15]. Moreover, Garg et al. focuse on CI/CD and auto-deployment of ML models using MLOps on Kubeflow [16].

To this end, previous work offers solutions for important but separate lifecycle functions such as orchestration, validation, documentation, and monitoring, while existing MLOps architectures mainly focus on training, deployment, or distributed execution independently. This paper does not claim novelty in these capabilities individually. Instead, its contribution is architectural, offering a lifecycle-oriented architecture that treats data-centric services, model-centric services, trustworthiness mechanisms, and observability as interoperable first-class services. Thus, the novelty of AI-DAPT lies in lifecycle unification and trustworthiness-by-design rather than in a new standalone MLOps or data-centric tool.

III. Methodology

A. Data and AI Pipeline Lifecycle

As ML systems move from prototypes to operational services, the dominant risks often arise from data rather than from the ML algorithm itself. Data-centric AI has shown that data quality, coverage, maintenance, provenance, and documentation are central to robustness, auditability, and long-term trustworthiness [4], [5], [17]. At the same time, production ML pipelines accumulate hidden technical debt through implicit assumptions, evolving schemas, feedback loops, and ad-hoc integration logic, making failures harder to diagnose and systems harder to maintain [7]. Although existing platforms provide support for orchestration, experiment tracking, and deployment [1], [2], capabilities such as dataset documentation, large-scale data quality verification, and post-deployment observability are treated as separate tools [6], [14].

These limitations motivate AI-DAPT as a unified reference architecture that brings data-centric engineering and model-centric lifecycle management into a single AIOps-oriented platform. At a high level, AI-DAPT adopts a data-centric, adaptable lifecycle view of automated data and AI pipelines spanning data design, data sculpting, data generation, model delivery, and post-deployment optimization. Its phases are structured so that outputs from one stage feed the next while remaining compatible with iterative operation in production. In practice, this lifecycle can be instantiated either as exploratory pipelines executed once or a few times, or as operational workflows running on a schedule, processing streams, or triggered by events. Next, we outline the stages of the AI-DAPT lifecycle shown in Fig. 1.

Data Design. This phase focuses on framing the business problem and selecting suitable datasets under domain expert guidance. Data are then acquired through automated harvesting from available sources, followed by exploratory analysis that reveals distributions, patterns, and relationships. In parallel, datasets should be documented through metadata that supports discovery, reuse, and traceability. Finally, initial quality and

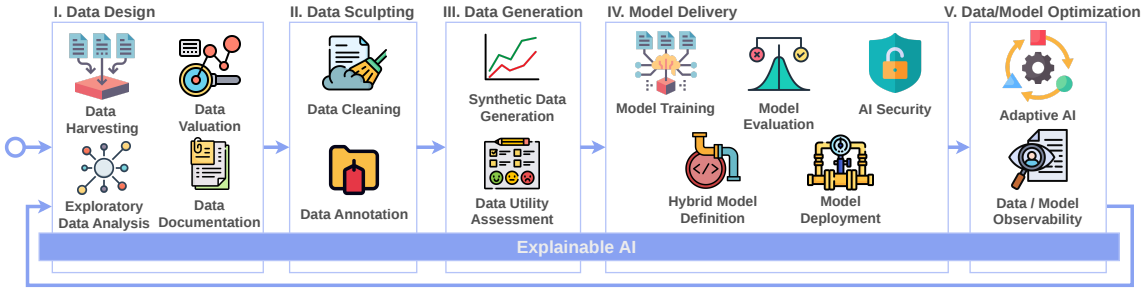

Fig. 1. AI-DAPT: AI and Data Pipeline Lifecycle

fitness checks can be performed through semantic labeling and data valuation activities so that downstream model development is grounded on purposeful and well-assessed data.

Data Sculpting. The second phase improves the representativeness and readiness of the data for learning. It includes annotation, selection, and cleaning steps that address missing values, noise, inconsistencies, and semantic gaps. These operations commonly rely on AI-based techniques but retain human oversight for validation and for handling domain-specific constraints. The intended outcome is an AI-ready dataset that is aligned with the chosen data model and is suitable for the requirements of the target learning task.

Data Generation. When real data are limited, inaccessible, or constrained by privacy and fairness considerations, the lifecycle should incorporate synthetic data generation to supplement or replace the original data. This phase is paired with data utility assessment to verify that generated samples preserve the most relevant properties of the original distribution and remain fit for downstream training and evaluation.

Model Delivery. This phase covers the progression from model definition to deployment. Models should be configured, trained, and evaluated, and the resulting artifacts should be prepared for operational use. AI-DAPT also supports hybrid science-guided development patterns, where first-principles models can inform, constrain, or complement data-driven models, and where efficiency-oriented choices, such as sparse modelling, can reduce computational demands in deployments.

Data/Model Optimization. This phase focuses on sustaining performance and trustworthiness after deployment. Data observability monitors pipeline health and data distribution changes to detect drift and quality regressions, while model observability tracks predictive performance and operational signals to identify degradation. These metrics can trigger adaptive actions, including retraining or other corrective updates, under policies that allow users to intervene when needed.

Explainable AI is a horizontal capability applied throughout the lifecycle. It increases the transparency of data and model decisions, supports debugging, and provides actionable insights for human-in-the-loop governance across all phases.

B. Requirements and Design Choices

The frame of the aforementioned lifecycle was established through a structured user-needs and requirements engineering process. Specifically, user needs were elicited from 4 demonstrators through 16 semi-structured interviews and 3 questionnaire rounds. These inputs were then analyzed and

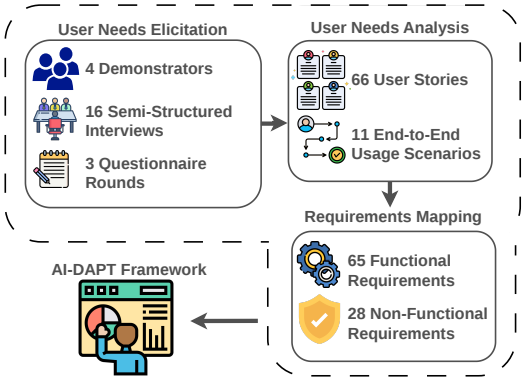

Fig. 2. Lifecycle from User Needs to Technical Requirements

consolidated into 66 user stories and 11 end-to-end usage scenarios, which were subsequently mapped into 65 functional and 28 non-functional requirements that informed the design of the AI-DAPT framework, as illustrated in Fig. 2.

Functional Requirements: Functionally, AI-DAPT must support the full data and AI pipeline lifecycle with explicit human oversight across the five AI-DAPT phases. This translates into the ability to compose pipelines from reusable operators, configure execution, and persist intermediate artifacts and metadata to enable reproducible reruns. A central requirement is end-to-end operational transparency across pipeline execution. AI-DAPT must provide logging, real-time progress monitoring, and notifications covering dataset status updates, pipeline stage transitions, failures, and optimization actions applied during execution. It must also support production-oriented execution modes, including scheduled, event-triggered, and near-real-time execution when required by the application context. Interoperability and user-centric experimentation are equally required for adoption. The platform must also support integration with existing software platforms and data sources through exposed APIs, enable data importing from external platforms and services, and support interactive experimentation through computational notebooks for dataset exploration and prototyping. Finally, AI-DAPT must expose capabilities that connect data decisions to model outcomes, including explainability and optimization suggestions, comparison across model trials, and monitoring at multiple points in the pipeline.

Non-Functional Requirements: Non-functional requirements capture the quality attributes needed to sustain long-running and evolving pipelines across domains. Accordingly, AI-DAPT must remain maintainable and well documented, while efficiently handling and storing large datasets with acceptable

performance under scale and concurrent access. Moreover, reliability requires service continuity and rapid failure recovery through failover support, backups, and consistency mechanisms. Security, privacy, and governance are treated as first-class capabilities, requiring controlled registration and access management, end-to-end encryption for data in transit and at rest, and traceability through user action logging and audit-relevant operational records to support by-design data protection, consent, transparency, and data flow accountability.

Design Choices: These requirements motivate a *service-oriented architecture* in which capabilities are delivered as **loosely-coupled microservices**. This decomposition enables independent evolution of data management, model lifecycle, and governance services, while reducing integration overhead and limiting maintenance costs caused by tight coupling. The AI-DAPT lifecycle framework (Sec. III) is reflected in the reference architecture by managing data-centric capabilities around the data design, sculpting, and generation phases, and by organizing model-centric capabilities around model delivery and continuous optimization. Additionally, observability and adaptive functions materialize the optimization phase in operation, while platform management provides shared functionalities, including security, access control, and interoperability across the full stack. To support reproducibility and traceability, AI-DAPT relies on a persistent artifact and metadata backbone that records datasets, pipeline specifications, execution traces, and model artifacts throughout the lifecycle. Transparency and continuous improvement are supported by embedded observability through continuous monitoring, logging, and notifications. Finally, secure-by-design mechanisms, including role-aware access control, encryption at rest and in transit, and auditable action logging, are integrated across services to satisfy privacy, security, and accountability.

IV. The AI-DAPT Reference Architecture

The proposed AI-DAPT architecture (Fig. 3) defines an end-to-end platform that unifies data-centric engineering and model-centric lifecycle management within a single service-oriented stack. Users interact with the platform through the **AI-DAPT User Interface (UI)** or via **exposed APIs** with dedicated documentation. From this entrypoint, admins configure the platform, which coordinates loosely-coupled microservices spanning the full lifecycle, from raw data acquisition to deployed and continuously monitored AI services, while preserving provenance and operational traces across all stages.

The architecture follows a multi-layered design, where the lowest layer is the **Scalable Storage Services** layer. It comprises a set of high-performance distributed storage systems, each optimized to persist the long-lived artifacts produced and consumed by the platform. Specifically, it stores datasets, pipelines, system logs, trained models, experimental metadata, results, explanations, and the operational state required for monitoring and runtime management. Essentially, it serves as the platform's record system, supporting reliable and reproducible platform operation.

On top of the storage services, AI-DAPT provides two tightly coupled layers, **Data Pipeline Services** and **Data Lifecycle Management Services**, which together support dataset ingestion, preprocessing, indexing, and semantic enrichment. **Data Pipeline Services** operationalize the ingestion and preparation of incoming data. They comprise (i) the *Data Harvester*, which acquires data either *at rest* (batch files) or *in motion* (near real-time streams); (ii) the *Data Annotation Engine*, which performs semantic annotation at the feature-schema level and applies any required harmonization; and (iii) the *Data Cleaning Engine*, which executes user-defined rules and/or automated ML-based cleaning strategies to improve data quality. While the Data Pipeline Services handle the flow of incoming datasets, the **Data Lifecycle Management Services** act as the platform's intermediary layer that manages and exposes stored data consistently across pipelines. The *Semantics Reconciliation Manager* on the other hand, supports dataset ingestion by coordinating cross-domain semantic models, ensuring integrity and validity while remaining adaptable to evolving stakeholder requirements. The *Data Documentation Engine* materializes these semantic models by extracting and maintaining dataset- and feature-level metadata, enabling traceability and governance. Finally, the *Data Features Toolkit* assists users during data preparation by supporting feature engineering and transformation, preparing them for training and analytics to improve model effectiveness.

The next two component groups, namely **Data-AI Execution Services** and **Data-AI Pipeline Monitoring Services**, support the data and ML processing part of the platform. The **Data-AI Execution Services** layer orchestrates the end-to-end execution of data and AI pipelines by coordinating service invocations and enforcing reliable handling of intermediate and final artifacts. At its core, the *Pipeline Execution Engine* dispatches submitted pipelines to external execution environments such as Kubernetes [18], Airflow [19], Spark [20], and MLflow [3], monitors execution and persists results, metrics, and logs. The *Pipeline Manager* supports the design and lifecycle of data and AI workflows through reusable operators, service integration, and a catalog of AI/ML algorithms, while the engine uses the resulting Directed Acyclic Graph (DAG) to allocate resources and schedule execution. Finally, the *Interactive Experimentation Engine* provides an interactive environment for rapid prototyping and experimentation via computational notebooks, enabling iterative development.

To support execution services, the **Data-AI Pipeline Monitoring Services** provide continuous, real-time oversight of deployed pipelines and production models. Within this layer, the *Data Observability Service* monitors end-to-end data pipeline runs, tracking freshness, completeness, and operational health, while detecting anomalies and triggering alerts when violations or unexpected patterns occur. In parallel, the *Model Observability Service* tracks model behavior over time by monitoring predictive performance and operational behavior, identifying distribution shifts and drift, and recommending corrective actions such as retraining, recalibration, etc. Building on these metrics, the *Adaptive AI Services* help sustain

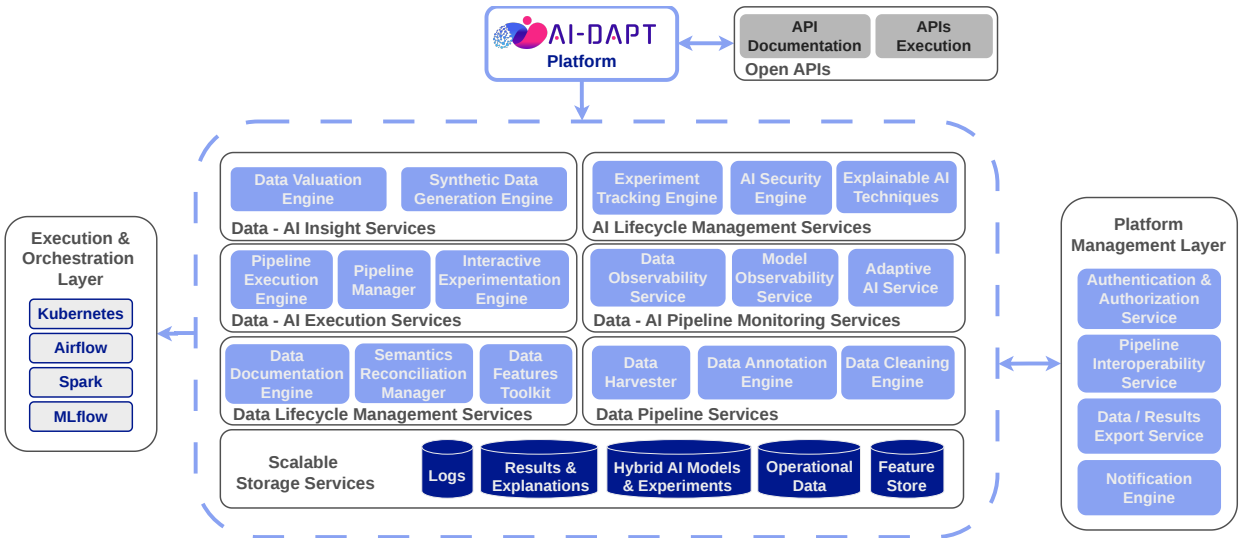

Fig. 3. Reference Architecture Overview

robustness in production via human-in-the-loop strategies, combining observability metrics with user-defined policies.

At the highest layers, AI-DAPT exposes user-centric capabilities through the **Data-AI Insight Services** and **AI Lifecycle Management Services**. The **Data-AI Insight Services** help users assess and improve datasets prior to model development. Specifically, the *Synthetic Data Generation Engine* supports the generation and evaluation of synthetic data using generative AI models. In parallel, the *Data Valuation Engine* estimates feature-level utility by quantifying feature quality and contribution, identifying bias and other issues that may affect training, and suggesting mitigation actions for data preparation. Complementing data-centric insights, **AI Lifecycle Management Services** support the design, validation, and operational governance of AI models, while the *Experiment Tracking Engine* captures training runs, configurations, metrics, and artifacts, enabling systematic comparison, reproducibility, and iterative optimization. The *AI Security Engine* strengthens robustness by assessing pipelines and models for security risks and supporting the detection of adversarial behaviors that compromise reliability. Lastly, the *Explainable AI Techniques* component offers a portfolio of interpretability methods that increase the transparency of trained hybrid AI models, offering actionable explanations that facilitate debugging, validation, and compliance-oriented governance.

Finally, the **Platform Management Services** provide cross-cutting capabilities across the stack, allowing all components to rely on shared governance and operational primitives. The *Authentication & Authorization Service* manages secure identities and fine-grained UI/API permissions, while the *Notification Engine* delivers alerts on pipeline progress, failures, and policy-relevant conditions. The *Data/Results Export Service* enables authorized extraction and sharing of artifacts, and the *Pipeline Interoperability Service* supports integration with external tools and AI-DAPT-compliant solutions through standardized interfaces and translation mechanisms.

Beyond the logical organization of services, the reference architecture is designed for infrastructure portability through containerization. AI-DAPT is currently deployed on a self-hosted cluster, where microservices run as containers, but the same architecture can be extended to managed cloud platforms, such as Amazon EKS. Orchestration, storage, observability, and access control remain decoupled from the application logic, enabling deployment across private, public, or hybrid cloud environments while supporting elasticity, multi-tenancy, fault isolation, and cloud-scale operation.

A. Current Realization and Technology Stack

This section presents the current realization of the AI-DAPT architecture by mapping the services introduced in Sec. IV to concrete technologies and implementation choices. Rather than reintroducing the architectural components, the focus here is on how these services are instantiated, integrated, and deployed within a microservice-based technology stack.

1) Scalable Storage Services:

The **Scalable Storage Services**, introduced in Section IV as the persistence backbone of AI-DAPT, are implemented using a polyglot storage architecture. A REST-based core endpoint provides a uniform access layer for platform services and dispatches requests to specialized storage backends according to artifact type. The implementation combines MinIO [21] for object and file artifacts, InfluxDB [22] for time-series data and monitoring traces, MongoDB [23] for semi-structured records, and Virtuoso [24] for RDF-based metadata and provenance. This design preserves backend-specific optimizations while exposing a consistent REST integration across the platform.

2) Data Pipeline Services:

Data Harvester: is the ingestion entrypoint, supporting heterogeneous data acquisition from files, databases, APIs, and streaming sources. It provides source selection, metadata entry, REST-based uploads, parallel ingestion, and validation of incoming data for downstream services. The component supports batch uploads in CSV, JSON, Excel, and Parquet, direct database queries, REST-based collection, and real-time ingestion. After ingestion, datasets are stored in the Scalable Storage Services, while metadata are maintained in the Data

Documentation Engine, making the Data Harvester the bridge between external sources and the AI-DAPT data lifecycle.

Data Annotation Engine: is implemented as a hybrid manual and automated annotation service operating on domain models supplied by the Semantics Reconciliation Manager. It supports structure-level alignment against a given model, as well as point-wise and interval-based annotation. The implementation includes a UI for manual annotation, AI-assisted labeling, and rule-based automatic annotation, whose settings can be saved for pipeline reuse. The service retrieves data models from the Semantics Reconciliation Manager and data files plus annotation configurations from the Scalable Storage Services, and writes back reconciled data, annotations, and configurations.

Data Cleaning Engine: is a preprocessing component for multiple dataset types, with particular support for tabular and time-series data. For time-series inputs, it provides anomaly detection, missing-value imputation, and repair workflows combining anomaly masking with imputation. The service exposes both REST and graphical interfaces for programmatic and interactive use, allowing users to configure methods and inspect anomalies and repaired data. To support traceability and reproducibility, it maintains an audit log of applied operations and propagates cleaned datasets, execution logs, and metadata to the Scalable Storage Services.

3) Data Lifecycle Management Services:

Data Documentation Engine: is the metadata and provenance backbone, providing catalog, discovery, and traceability capabilities for datasets and AI artifacts across the platform. It is built on the open-source Piveau stack [25], uses DCAT-AP [26] as its core metadata standard, and supports domain-specific extensions through Shapes Constraint Language (SHACL). Internally, it combines RDF-based metadata persistence in Virtuoso, search and indexing through Elasticsearch, provenance recording, and a catalog interface. It exposes JSON-based retrieval APIs for integration with other AI-DAPT components and acts as an intermediary layer for artifact discovery and traceability before direct storage access. Metadata are supplied by services such as the Data Harvester and Synthetic Data Generation Engine, while downstream components such as the Pipeline Manager consume the resulting metadata.

Semantics Reconciliation Manager: is the semantic governance layer of AI-DAPT, ensuring that shared data models remain valid, versioned, and consistently consumable across services through ontology-based semantic interoperability approaches [27], [28]. It provides endpoints for semantic model upload, retrieval, update, deletion, validation, and UI-based editing. Validated models are stored with metadata and version information in Storage and exposed through a dedicated serving layer. The service delivers JSON representations of semantic models to other components, particularly the Data Annotation Engine, supporting consistent schema reconciliation and interoperable data transformations across the platform.

Data Features Toolkit: supports feature selection, extraction, and preprocessing, including dimensionality reduction, encoding, transformation, aggregation, and imputation. At its core is an interactive Data Features UI Designer for dataset retrieval, configuration loading, feature engineering, metadata updates, and logging. The backend relies on Pandas [29], integrates with scikit-learn [30], and supports libraries such as Featuretools [31]. For scalability, it can use distributed execution through Dask [32] or Modin [33], while long-running computations are offloaded to Celery workers [34]. The component connects to Storage for artifacts and to the Data Documentation Engine for metadata registration.

4) Data-AI Execution Services:

Pipeline Execution Engine: is the runtime orchestration layer, responsible for translating DAG-based workflow specifications into executable deployments under explicit performance requirements. Its architecture includes a FastAPI-based REST API, an Orchestrator, a Pipeline Delivery module, and a Pipeline Monitoring module. Upon receiving DAGs from the Pipeline Manager, the engine plans execution, generates deployment descriptors, and dispatches pipeline stages to execution substrates such as Kubernetes, Apache Airflow, and Apache Spark, bridging pipeline design with AI-DAPT's MLOps-oriented execution stack. The monitoring module integrates Prometheus [35] to collect runtime metrics and support performance-aware reactions during execution. Execution metrics, logs, and status updates are propagated to the Scalable Storage Services and the Notification Engine.

Pipeline Manager: is implemented as the Data Analytics and Visualization Environment (DAVE [36]), and serves as AI-DAPT's main no-code environment for workflow authoring. Built on Apache Airflow, DAVE provides DAG-based workflow definition, scheduling, dependency management, and monitoring. Computational logic is packaged as reusable Operators grouped into Connectors, Transformers, and Analytics. The runtime integrates Apache Spark for distributed processing, Pandas for dataframe transformations, scikit-learn for preprocessing and analytics, and MLflow for experiment tracking. Its UI supports drag-and-drop workflow construction, operator configuration, persistence, and extensions, including Python-based or API-invoked operators, and human-in-the-loop tasks. Datasets, logs, metadata, configurations, and models are exchanged with the storage layer, while DAG specifications are forwarded to the Pipeline Execution Engine for deployment and execution.

Interactive Experimentation Engine: is a notebook-centric environment for exploratory analysis, data processing, and model development, built around JupyterHub [37] for multi-user access and reproducible sessions. Users can run Python, R, and Julia notebooks with ML-oriented dependencies, while MLflow-backed access patterns support experiment management and artifact access. A REST middleware integrates the UI, JupyterHub environments, and the rest of the platform by managing notebook sessions and retrieving models and datasets from Storage. This design supports collaborative experimentation, centralized environment management, and seamless access to AI-DAPT datasets and artifacts.

5) Data-AI Pipeline Monitoring Services:

Data Observability Service: is a rule-driven monitoring com-

ponent for assessing data quality and operational health across the pipeline lifecycle. It consumes datasets together with observability configurations specifying monitored attributes, thresholds, and rules, and produces metrics, logs, and alerts related to data quality, drift, anomalies, and overall data health. The service integrates with the Scalable Storage Services for dataset and configuration retrieval and can operate either standalone or as a pipeline step triggered by the Pipeline Manager. Metrics and reports are exposed through REST APIs, alerts are propagated to the Notification Engine via Kafka [38], and operational traces are stored back in the storage layer to support traceability and downstream adaptation.

Model Observability Service: is implemented as a multi-source monitoring stack centered on a Python-based Model Observability Engine, forming the model monitoring layer of AI-DAPT workflows. The service combines real-time prediction streams, historical performance records, observability values, and user-defined thresholds to compute drift indicators, quality metrics, statistics, and outliers. Historical data and thresholds are retrieved through REST APIs, while live model output is ingested via Kafka, analyzed, and republished through a Kafka producer. The resulting metrics are exposed through a visualization dashboard and are forwarded to the Adaptive AI Service to support timely intervention, retraining, and model adaptation. Metrics, thresholds, and logs are persisted in the Scalable Storage Services.

Adaptive AI Service: is a human-in-the-loop adaptation layer that translates observability evidence into retraining and rebuilding decisions. Its architecture includes REST APIs for AI models, pipelines, historical and training data, together with components for rule definition, adaptation, inference, retraining, and assessment. For real-time monitoring, the service consumes model and data performance streams through Kafka. The adaptation engine enables MLOps and AutoML paradigms by updating Pipeline Manager DAG configurations and recommending alternative models and feature engineering strategies. It also leverages MLflow to track retraining runs and model versions, while persisting logs and reusable DAG versions in the storage layer. In this way, it acts as the main decision and control loop for adaptive AI within AI-DAPT.

6) Data-AI Insight Services:

Synthetic Data Generation: is a configurable service that generates tabular and time-series synthetic data. Its core component, the Synthetic Data Generator, executes Python scripts implementing methods such as Generative Adversarial Networks (GANs), variational autoencoders, and statistical sampling. The generator interacts with dedicated read endpoints for data, models, and metadata to access datasets, pretrained models, and metadata that guides the generation process. Generated datasets are stored along with their metadata in Scalable Storage Services, and can be discovered through the AI-DAPT catalog. Finally, its UI presents visual outputs for assessing realism and utility, while informing the user of completed tasks through the Notification Engine.

Data Valuation Engine: is implemented as an assessment service that quantifies dataset quality, representativeness, and fairness before learning. It can be invoked via its dedicated UI and supports feature importance, correlation analysis, and bias and fairness auditing modules. These modules are built on top of libraries such as scikit-learn for statistical and model-based assessment, SHAP for attribution-based explanations, and AI Fairness 360 [39] for bias detection across sensitive groups. Depending on the selected module, the service consumes datasets together with labels, model outputs, or probability scores, and produces rankings, correlation matrices, fairness reports, and other valuation artifacts stored in Storage.

7) AI Lifecycle Management Services:

Hybrid AI Model Engineering Engine: is realized as a set of Apache Airflow Python Operators, each encapsulating a hybrid or science-guided model as a Python function. These operators are integrated into DAVE and exposed through the dashboard, allowing hybrid model training and evaluation to be composed within larger AI-DAPT workflows. During execution, the component consumes datasets from the Scalable Storage Services, trains and evaluates hybrid models, and stores the resulting artifacts and training metadata in the storage layer. This operator-based design ensures compatibility with the orchestration stack used by the Pipeline Manager and the Pipeline Execution Engine, while enabling science-guided AI workflows within the broader AI-DAPT platform.

Explainable AI Techniques: are operator-based extensions that integrate explainability directly into AI-DAPT workflows. Each method is packaged as an Airflow Operator compatible with DAVE and executed through the Pipeline Execution Engine using the trained model and the required configuration. The component supports both global and local explanation workflows and incorporates established Explainable AI (XAI) methods such as SHAP for feature attribution and model interpretation. The produced explanation artifacts are stored in the Scalable Storage Services and exposed through an interactive visualization dashboard, enabling users to inspect multiple explanation views depending on the selected method and model. Thus, explainability becomes a first-class pipeline operation rather than an offline post-processing step.

Experiment Tracking Engine: is the experiment management and visual analytics service that supports the inspection and comparison of AI training runs. It ingests run-level statistics, artifacts, and metadata from Scalable Storage Services and exposes them through dashboards, enabling reproducibility, model selection, and experimentation transparency. The information collected allows users to analyze differences across runs and trace the evolution of the model over time. Additionally, the component operates as an MLflow-enabled tracking layer for experiment metadata and lifecycle observability, interfacing with the storage layer through REST APIs.

AI Security Engine: is the security and adversarial robustness service for AI models, supporting trustworthy deployment by assessing models and pipelines against threats such as poisoning, evasion, backdoor, and privacy-leakage attacks. It exposes REST APIs that can be invoked directly or via an Airflow operator integrated with the Pipeline Manager, enabling security assessment within workflows. The engine

retrieves trained models and artifacts from storage, executes the selected analysis, and stores vulnerability reports, mitigation recommendations, and audit-relevant artifacts back in storage. In this way, it acts as a governance layer that enhances robustness, traceability, and secure lifecycle management.

8) Platform Management Services:

Authentication & Authorization service: is implemented around Keycloak [40] as the central identity and access management service of AI-DAPT. Internally, the Keycloak server provides authentication, session management, role-based authorization, user and group management, and protocol handlers for OIDC, OAuth2, and SAML. Persistent user and session information is stored in a dedicated database within the storage layer, while administrative management is carried out through a React and JavaScript-based Admin UI. All AI-DAPT services and external clients can request access tokens through REST-based interactions, and these tokens are then used to enforce access policies across pipelines, configurations, datasets, and exportable artifacts.

Notification Engine: is an event-driven user notification subsystem of AI-DAPT. Its internal architecture contains a Notification API for retrieval and state updates, such as marking notifications as seen, and a family of Notification Handlers including web-based, email, and push delivery handlers. Platform services publish JSON notification events to Kafka, where the notification handlers process and store them.

Data Results Export Service: is the controlled delivery layer for exporting datasets, model outputs, and other AI-DAPT artifacts to external users and applications. An Asset List Endpoint exposes a REST API for browsing assets, while a Node.js and Vue.js-based Asset Query Engine supports preview, filtering, and reusable export configurations. Data retrieval and delivery are handled through dedicated APIs, while integrations with the Data Documentation Engine and Storage support metadata retrieval, asset access, and configuration persistence.

Pipeline Interoperability Service: is created as the bridge between AI-DAPT and external analytics platforms. It exposes an Incoming Pipeline Execution Request REST API that accepts external requests, which are then processed by an Execution Resolver implemented in Node.js. The resolver transforms the incoming request into a DAG representation suitable for AI-DAPT execution and forwards it to a DAG Pushing Endpoint, which sends it to the internal pipeline runtime. The service also exposes a Result Read Endpoint so that external systems can retrieve results after execution, and a Logging Interface records requests and transformations for traceability.

V. Discussion

A. Answering the Research Question

As already discussed, existing AI and MLOps ecosystems are increasingly fragmented, with data engineering, model lifecycle management, observability, governance, and trustworthiness capabilities often handled through different isolated tools. This fragmentation complicates reproducibility, interoperability, and adaptive AI operations across the full lifecycle. Consequently, this article answers the following research question: *How can a service-oriented architecture support reproducible, observable, and governable AI pipelines across the full data and model lifecycle?* The AI-DAPT architecture addresses this question through a lifecycle-oriented microservice-based approach that unifies data- and model-centric operations while embedding observability, governance, provenance, and AI capabilities across a full pipeline lifecycle.

B. Match and Contributions

The main contribution of the AI-DAPT Architecture is not the introduction of yet another orchestration engine, experiment tracker, monitoring framework, or standalone trustworthy AI mechanism. Instead, it provides a unified lifecycle-oriented and practical framework that integrates data engineering, model operations, observability, explainability, and adaptive AI within an interoperable platform. Unlike conventional MLOps platforms focused mainly on training, deployment, and automation, AI-DAPT treats metadata, provenance, security, and human oversight as native cross-lifecycle services. This integration links data preparation, monitoring, retraining, and governance while preserving interoperability with common orchestration, storage, and experimentation tools.

In practice, the architecture supports heterogeneous operational settings where pipelines evolve continuously and require reproducibility, auditability, controlled adaptation, and management of emerging AI technologies. For example, observability services can identify data quality degradation or predictive drift during deployment, while the Adaptive AI Service can trigger retraining workflows or recommend updated feature engineering strategies under human supervision. Similarly, explainability and valuation services connect data decisions to downstream model behavior, improving transparency and supporting governance-aware AI operations.

Moreover, the proposed architecture is modular and extensible, enabling support for both current and future AI paradigms, although AI-DAPT currently focuses on data-centric and predictive AI workflows. In particular, its experimentation and observability capabilities can naturally handle emerging technologies and support not only conventional model training, but also the training and lifecycle management of generative models, including large language models (LLMs).

Additionally, the proposed architecture may introduce additional operational complexity compared to tightly integrated monolithic MLOps solutions. However, this complexity is mainly associated with the initial deployment and integration of the overall system. Once deployed, the architecture provides benefits in terms of modularity, extensibility, interoperability, and lifecycle traceability. In contrast, monolithic ML solutions often require users to repeatedly select, integrate, and adapt components for each new ML pipeline, which can increase long-term maintenance effort. Thus, the initial setup cost is acceptable when reproducibility, auditability, governed adaptation, and long-term maintainability are critical.

VI. Conclusion

In summary, AI-DAPT contributes a lifecycle-oriented reference architecture that unifies data-centric and model-centric operations with trustworthiness capabilities in a microservice-based platform. By structuring the platform around five iterative phases and embedding metadata, lineage, observability, explainability, security, and human oversight as native services, the architecture provides a blueprint for reproducible, observable, and governable AI pipelines. Rather than replacing existing tools, AI-DAPT is designed to interoperate with widely used storage, orchestration, monitoring, and experimentation technologies, supporting heterogeneous deployment settings.

Finally, although this paper focuses on describing the AI-DAPT architecture, future work will evaluate its usability within the project through four domain pilots and eleven scenarios. This evaluation will assess reproducibility, observability-driven adaptation, interoperability, and operational overhead under real deployment conditions. The pilots cover the electricity market, maintenance operations in the aviation manufacturing industry, human-centered industrial robotics with a focus on cognitive ergonomics, and personalized healthcare. Through these use cases, we will also examine practical trade-offs, including how operators perceive system complexity, assess deployment overhead, and address interoperability challenges when integrating existing code.

Acknowledgment

This work is supported by the EU Commission through AI-DAPT project (HORIZON-CL4-2023-HUMAN-01-01, Grant Agreement: 101135826). Language refinements were performed at the sentence level using ChatGPT. All original content and ideas are solely those of the authors.

References

[1] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, C. Y. Koo, L. Lew, C. Mewald, A. N. Modi, N. Polyzotis, S. Ramesh, S. Roy, S. E. Whang, M. Wicke, J. Wilkiewicz, X. Zhang, and M. Zinkevich, "Tfx: A tensorflow-based production-scale machine learning platform," in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: ACM, 2017, p. 1387–1395.

[2] J. de la Rúa Martínez, F. Buso, A. Kouzoupis, A. A. Ormenisan, S. Niazi, D. Bzhalava, K. Mak, V. Jouffrey, M. Ronström, R. Cunningham, R. Zangis, D. Mukhedkar, A. Khazanchi, V. Vlassov, and J. Dowling, "The hopsworks feature store for machine learning," in *SIGMOD '24*. New York, NY, USA: ACM, 2024.

[3] A. Chen, A. Chow, A. Davidson, A. DCunha, A. Ghodsi, S. A. Hong, A. Konwinski, C. Mewald, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, A. Singh, F. Xie, M. Zaharia, R. Zang, J. Zheng, and C. Zumar, "Developments in mlflow: A system to accelerate the machine learning lifecycle," in *Proc. DEEM*. New York, NY, USA: ACM, 2020.

[4] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. M. Wallach, H. D. III, and K. Crawford, "Datasheets for datasets," *Commun. ACM*, vol. 64, no. 12, pp. 86–92, 2021.

[5] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, "Model cards for model reporting," in *FAT**. ACM, 2019, pp. 220–229.

[6] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Bießmann, and A. Grafberger, "Automating large-scale data quality verification," *Proc. VLDB Endow.*, vol. 11, no. 12, pp. 1781–1794, 2018.

[7] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015.

[8] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine learning operations (mlops): Overview, definition, and architecture," *IEEE Access*, 2023.

[9] N. Janbi, I. Katib, and R. Mehmood, "Distributed artificial intelligence: Taxonomy, review, framework, and reference architecture," *Intelligent Systems with Applications*, vol. 18, p. 200231, 2023.

[10] A.-D. Team, "Ai-dapt project," 2026, https://www.ai-dapt.eu/.

[11] E. Caveness, P. S. G. C., Z. Peng, N. Polyzotis, S. Roy, and M. Zinkevich, "Tensorflow data validation: Data analysis and validation in continuous ml pipelines," in *Proc. 2020 ACM SIGMOD Int. Conf. on Management of Data*, ser. SIGMOD '20. NY, USA: ACM, 2020.

[12] F. Bachinger, L. Ehrlinger, G. Kronberger, and W. Wöss, "Data validation utilizing expert knowledge and shape constraints," *J. Data and Information Quality*, vol. 16, no. 2, Jun. 2024.

[13] H. Chen and S. Schelter, "Towards automated task-aware data validation," in *Proc. DEEM*. New York, NY, USA: ACM, 2025.

[14] S. Shankar and A. G. Parameswaran, "Towards observability for production machine learning pipelines," *Proc. VLDB Endow.*, 2022.

[15] P. Notaro, J. Cardoso, and M. Gerndt, "A survey of aiops methods for failure management," *ACM Transactions on Intelligent Systems and Technology*, vol. 12, no. 6, pp. 81:1–81:45, 2021.

[16] S. Garg, P. Pundir, G. Rathee, P. Gupta, S. Garg, and S. Ahlawat, "On continuous integration / continuous delivery for automated deployment of machine learning models using mlops," in *Proc. 2021 IEEE 4th Int. Conf. on Artificial Intelligence and Knowledge Engineering*, 2021.

[17] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu, "Data-centric artificial intelligence: A survey," *ACM Comp. Surv.*, 2025.

[18] Linux Foundation, "Kubernetes," 2026, https://kubernetes.io/.

[19] A. S. Foundation, "Apache airflow," 2026, https://airflow.apache.org/.

[20] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX NSDI*, 2012.

[21] MinIO, "Minio documentation," 2026, https://docs.min.io/.

[22] InfluxData, "Influxdb," 2026, https://docs.influxdata.com/.

[23] MongoDB, "Mongodb," 2026, https://www.mongodb.com/.

[24] O. Erling and I. Mikhailov, "Rdf support in the virtuoso dbms," in *Networked Knowledge – Networked Media*, T. Pellegrini, S. Auer, K. Tochtermann, and S. Schaffert, Eds. Springer, 2009.

[25] P. Contributors, "piveau," 2026, https://www.piveau.de/en/.

[26] Publications Office of the European Union, "Dcat application profile for data portals in europe (dcat-ap)," 2026, https://bit.ly/4tCYvse.

[27] N. Bassiliades, M. Symeonidis, P. Gouvas, E. Kontopoulos, G. Meditskos, and I. P. Vlahavas, "Paasport semantic model: An ontology for a platform-as-a-service semantically interoperable marketplace," *Data & Knowledge Engineering*, 2018.

[28] D. Stefanidis, C. Christodoulou, M. Symeonidis, G. Pallis, M. D. Dikaiakos, L. Pouis, K. Orphanou, F. Lampathaki, and D. Alexandrou, "The icarus ontology: A general aviation ontology developed using a multi-layer approach," in *Proc. WIMS 2020*, 2020.

[29] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 56–61.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[31] F. Labs, "Featuretools docs," 2020, https://docs.featuretools.com/.

[32] M. Rocklin, "Dask: Parallel computation with blocked algorithms and task scheduling," in *Proc. 14th Python in Science Conf.*, 2015.

[33] D. Petersohn, D. Tang, R. Durrani, A. Melik-Adamyan, J. E. Gonzalez, A. D. Joseph, and A. G. Parameswaran, "Flexible rule-based decomposition and metadata independence in modin: A parallel dataframe system," *Proceedings of the VLDB Endowment*, vol. 15, no. 3, pp. 739–751, 2021.

[34] C. Contributors, "Celery," 2026, https://docs.celeryq.dev/en/stable/.

[35] P. Authors, "Prometheus," 2026, https://prometheus.io/.

[36] A. Grilo, P. Figueiras, B. Rêga, L. Lourenço, A. Khodamoradi, R. Costa, and R. Jardim-Gonçalves, "Data analytics environment: Combining visual programming and mlops for ai workflow creation," in *IEEE Int. Conf. on Engineering, Technology, and Innovation (ICE/ITMC)*, 2024.

[37] P. Jupyter, "Jupyterhub," 2026, https://jupyter.org/hub.

[38] A. S. Foundation, "Apache kafka," 2026, https://kafka.apache.org/.

[39] AI Fairness 360, "AI Fairness 360: Understand and Mitigate Bias in ML Models," https://ai-fairness-360.org/, 2024.

[40] K. Project, "Keycloak server," 2026, https://www.keycloak.org/.