

## Article

# A Human-Centric MLOps Blueprint for Visual Quality Control as a Service in an Open Access Platform

Krzysztof Wołk <sup>1,2,\*</sup>, Jacek Niklewski <sup>3</sup>, Marek S. Tatara <sup>3,4</sup>, Piotr Kopa-Ostrowski <sup>3,4</sup>  
and Michał Kopczyński <sup>3,4</sup>

<sup>1</sup> DAC.next Sp. z o.o., 80-309 Gdansk, Poland

<sup>2</sup> Social Academy of Science, 00-842 Warszawa, Poland

<sup>3</sup> DAC.Digital S.A., 80-309 Gdansk, Poland

<sup>4</sup> Department of Robotics and Decision Systems, Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, 80-233 Gdansk, Poland

\* Correspondence: krzysztof.wolk@dacnext.com

## Abstract

Visual quality control increasingly relies on machine learning (ML), but reliable production adoption requires a governed data-model-service lifecycle rather than an isolated detector. This article presents an implementation-grounded Open Access Platform (OAP) blueprint and audit protocol for visual quality control as a service. The contribution is not a new detector backbone. It is a reproducible governance pattern that connects ontology-based data validation, Neo4j knowledge-graph traceability, Eclipse Arrowhead service discovery, human-in-the-loop annotation, dataset and model registries, operational quality gates, and controlled model adaptation. The platform context is supported by implemented OAP tools for battery carbon-footprint analysis and production planning, while the visual inspection modules are reported at the notebook level. The benchmark protocol uses leakage-safe splits and OK samples with empty label files, separates the MVTec AD augmentation corpus from the unlearning stress-test corpus, and requires FP-on-OK gating with a retrained post-removal reference. The resulting architecture supports semantic traceability, controlled promotion and rollback, human accountability and Industry 5.0-aligned industrial AI governance without overstating the completed evidence. Compared with existing MLOps architectures, OAPs add inspection-specific semantic enforcement, knowledge-graph lineage, and governed adaptation.

**Keywords:** MLOps; Open Access Platform; visual quality control; benchmark protocol; knowledge graphs; ontology; human-in-the-loop; synthetic defects; machine unlearning; Eclipse Arrowhead; Industry 5.0



Academic Editor: Alexander Gegov

Received: 21 May 2026

Revised: 19 July 2026

Accepted: 27 July 2026

Published: 31 July 2026

**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article

distributed under the terms and

conditions of the [Creative Commons](#)

[Attribution \(CC BY\) license](#).

## 1. Introduction

Machine learning is now widely used in visual inspection, but the move from a successful pilot to a plant-ready system remains difficult. A pilot detector can show promising accuracy under controlled conditions, yet fail in operation when the camera angle, illumination, material batch, or surface condition changes. In production ML, much of the long-term cost comes from the surrounding lifecycle of data, labels, monitoring, validation, deployment, rollback, and maintenance, not from the trained model alone [1,2].

Visual quality control therefore needs to be engineered as an operating system for data and models, not only as an object-detection experiment. Industrial inspection data are vulnerable to dataset shift, where training and operating distributions diverge [3].

Wider computer-vision studies also show that image classifiers can generalize poorly when evaluation conditions differ from original training distributions [4]. In manufacturing, this issue is intensified by changing surface texture, tooling, materials, camera settings, and inspection rules.

Another practical constraint is labelling. Defect annotation requires domain knowledge from quality engineers, operators or inspectors. Labelling every frame manually is expensive and can delay deployment. Interactive and human-in-the-loop machine learning is therefore important because it can reduce repetitive work while keeping humans responsible for quality decisions [5,6]. Active learning and semi-supervised learning can prioritize uncertain cases and reuse unlabeled data, but these techniques must be connected to auditability and quality gates rather than used as opaque automation [7,8].

The research gap addressed here is therefore not the design of another detector backbone. The central problem is how a visual inspection detector can be embedded into an auditable industrial platform in which data provenance, semantic consistency, service access, quality gates, monitoring, and human review remain explicit throughout the lifecycle. The methodological contribution is a lifecycle protocol. It specifies the governance objects that must be created, the evidence that must be attached to each object, and the gate decisions that determine whether a model can be promoted, revised or rolled back. In practical terms, these objects include product and ROI metadata, dataset and model versions, taxonomy entries, monitoring events, and human gate decisions. This distinction is important because many vision papers optimize a model, whereas production inspection also requires controlled data intake, registry discipline, rollback paths, and operational false-alarm limits.

The contribution is fourfold. First, the article documents implemented OAP use cases that demonstrate the semantic and service infrastructure required for governed industrial AI. Second, it defines a complete visual quality-control MLOps pipeline with dataset and model registries, audit logs, quality gates, and monitoring. Third, it formalizes a benchmark and promotion protocol that links data manifestation, train/validation/test separation, class-level evaluation, false-positive-on-OK measurement, and registry-backed gate decisions. Fourth, it incorporates two notebook-level adaptation modules as governed extensions: NanoBanana synthetic defect generation for rare or imbalanced defects, and approximate machine unlearning through restricted fine-tuning for class or data removal. The combined contribution is an operational method for connecting visual inspection, semantic traceability, and service governance rather than a claim of detector-level novelty.

To make the evidence level transparent, the manuscript distinguishes three evidence levels: implemented OAP functions, notebook-level visual inspection modules, and future live-service deployment. This three-level evidence structure prevents overclaiming while still showing how the components form an end-to-end service architecture. It also makes explicit whether each claim is supported by the implementation evidence, by benchmark-protocol evidence, or retained as a target for future industrial validation.

#### *Related Work and Architectural Positioning*

General MLOps reference architectures that organize production machine learning around data engineering, experimentation, automated pipelines, registries, deployment, serving, monitoring, and role separation [9]. Industry-derived and manufacturing-focused studies have additionally emphasized OT/IT integration, edge constraints, data quality, drift management, security, and stakeholder-specific workflows [10–12]. Production platforms, such as TFX, operationalize validating, training, and serving pipelines at scale [13], while continuous-AI pipeline research structures data handling, model learning, software

development, and system operations [14]. The Arrowhead architecture contributes late-bound service discovery and authorization for industrial systems [15].

Against this background, the OAP proposal is not positioned as a replacement for existing industrial MLOps platforms. Its specific contribution is to combine established lifecycle functions with ontology-based validation and Neo4j lineage and then expose the governed visual quality-control service through Arrowhead. Product identities, ROIs, taxonomies, dataset versions, model versions, monitoring events, and human gate decisions are linked as auditable objects. The inspection-specific false-positive-on-OK gate and governed adaptation modules therefore complement, rather than duplicate, conventional orchestration and registry functions. The architectures reviewed in [9–14] provide strong lifecycle orchestration but do not report native support for ontology-enforced evolution of defect taxonomies, auditable synthetic defect injections, or production-critical gates, such as the false-positive rate on clean (OK) samples. Table 1 summarizes this architectural comparison and identifies where the OAP contribution differs from the cited MLOps approaches.

**Table 1.** Architectural comparison of OAP and the cited MLOps approaches.

| Capability                                     | General MLOps [9,13,14]                             | Industrial MLOps [10,11]                                 | Vision-Inspection Lifecycle [12]                | OAP                       |
|------------------------------------------------|-----------------------------------------------------|----------------------------------------------------------|-------------------------------------------------|---------------------------|
| Lifecycle orchestration                        | Explicit                                            | Explicit                                                 | Explicit                                        | Integrated                |
| Ontology-enforced evolution of defect taxonomy | NR                                                  | NR                                                       | NR                                              | Explicit                  |
| Knowledge-graph lineage                        | NR                                                  | NR                                                       | NR                                              | Explicit                  |
| Industrial service discovery and authorization | NR                                                  | Integration and security considered; native discovery NR | NR                                              | Explicit via Arrowhead    |
| FP-on-OK promotion gate                        | NR                                                  | NR                                                       | NR                                              | Explicit protocol         |
| HITL label provenance                          | Roles and workflows considered; label provenance NR | Stakeholders considered; label provenance NR             | Lifecycle roles considered; label provenance NR | Explicit                  |
| Auditable synthetic defect generation          | NR                                                  | NR                                                       | NR                                              | Auditable notebook module |
| Governed unlearning                            | NR                                                  | NR                                                       | NR                                              | Governed notebook module  |

Note: NR means that the capability is not reported as a native element of the cited architecture; it does not imply that the capability cannot be implemented as an extension.

## 2. Materials and Methods

### 2.1. Open Access Platform and Service Context

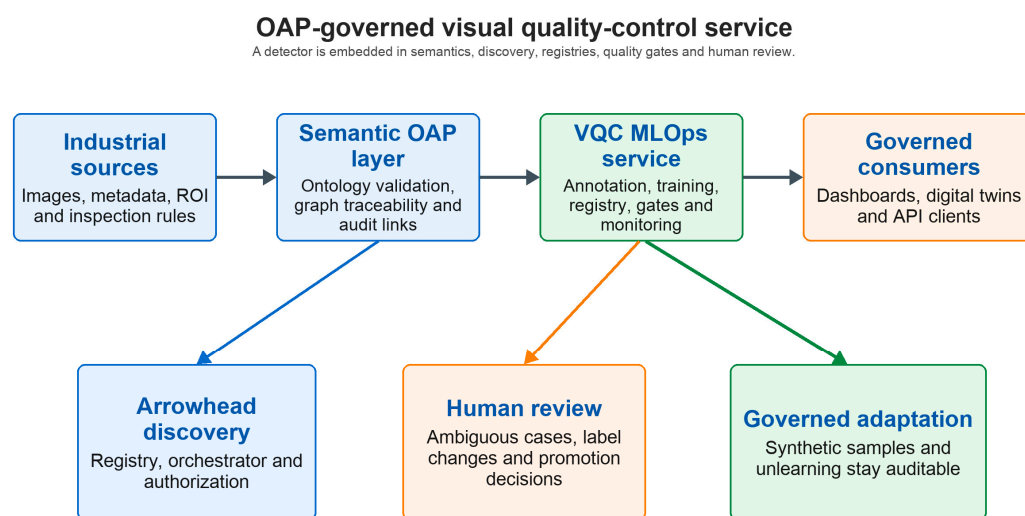
The OAP is used as the operational layer that connects industrial data, ontologies, knowledge graphs, application services, and external consumers. Rather than treating each AI model as a stand-alone tool, the platform provides a shared environment for data representation, validation, storage, visualization, service access, and governance.

In the implemented OAP use cases, industrial data are represented through explicit semantic structures. Ontologies describe the expected domain concepts and relationships. The uploaded data are validated against these structures before storage. This is important for visual quality control because inspection models depend on consistent metadata, in-

cluding the product type, image source, ROI, defect taxonomy, label version, model version, and deployment status.

Knowledge graphs are suitable for this role because they represent entities and relationships in a form that can be searched, visualized, and linked across services [16]. In the OAP setting, this supports traceability between industrial assets, datasets, models, monitoring signals, and decisions. The visual quality-control service can therefore be added as another platform service rather than as an isolated script or notebook.

The platform also supports service integration. The Arrowhead demonstration shows that OAP services (Figure 1) can be registered, discovered, and accessed through a standardized Industry 4.0 service-discovery mechanism [17]. This matters for MLOps because model consumers should not be forced to hard-code unstable endpoint locations. Dynamic discovery and authorization allow external systems, such as digital twins, analytics dashboards, or planning tools, to discover services at runtime while access remains governed.

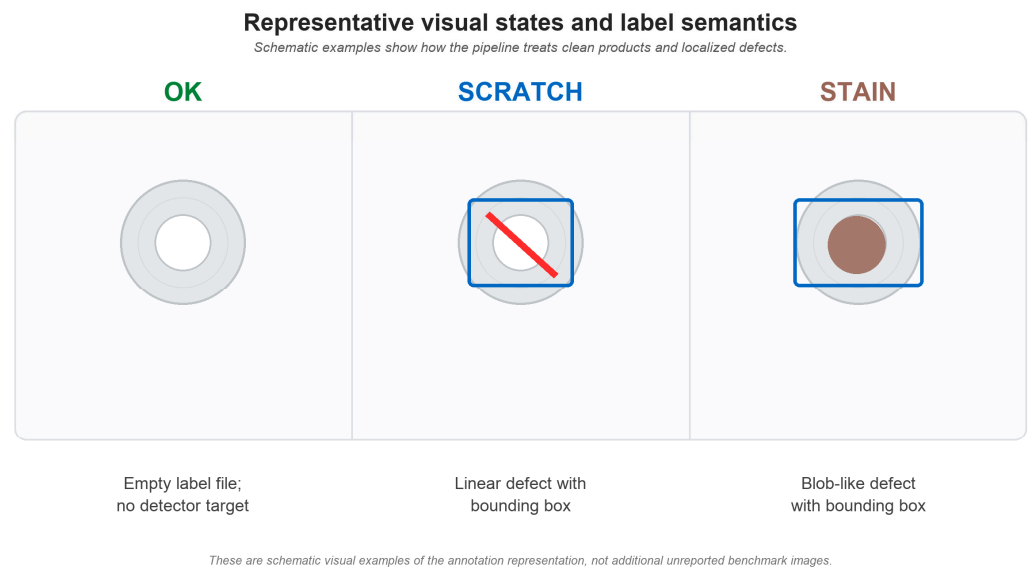


**Figure 1.** OAP-governed visual quality-control service architecture. The figure separates data sources, the semantic OAP layer, the VQC MLOps service, governed consumers, Arrowhead discovery, human review, and adaptation modules without crossing annotations. It also identifies the governance objects carried from ROI metadata through registry records to gate decisions.

## 2.2. Visual Quality-Control Pipeline

The visual quality-control service (Figure 2) follows the same OAP pattern. It consumes images or video-derived frames, product metadata, ROI definitions, annotations, and dataset manifestations. It produces trained models, evaluation reports, prediction outputs, monitoring signals, and audit records. Its outputs can be connected to other OAP services, including monitoring views for production status and dashboards that support operational decisions.

The service focuses on surface-level visual defects that can be localized through bounding boxes and, where available, masks. The Task 1a augmentation notebook uses metal\_nut inspection data from the MVTec Anomaly Detection dataset, a benchmark for real-world industrial inspection scenarios [18]. In Task 1a, the target states are OK, SCRATCH, and STAIN. OK is treated as the absence of a defect: OK images have empty label files and are used for false-positive analysis rather than as a detection object. The Task 1b unlearning notebook is reported separately as a class-removal stress test with its own larger corpus and taxonomy; it is not presented as the same MVTec split.



**Figure 2.** Representative annotation states used by the pipeline. The OK state intentionally has no detection target, whereas SCRATCH and STAIN are localized with defect labels and bounding boxes. The examples are schematic visual representations of the label semantics.

This design choice is important for industrial deployment. A clean product should not require the model to detect an OK object; therefore, for OK samples, the correct output is the absence of defect boxes. The false-positive-on-OK rate therefore becomes a critical production metric because excessive false alarms increase the human review load and reduce trust. The service evaluates standard detection metrics while also tracking operational metrics, such as false positives on clean frames, latency, review burden, and stability under changed conditions.

### 2.3. MLOps Pipeline and Governance Controls

The operational MLOps backbone contains nine stages: image or video acquisition with metadata, frame selection, ROI definition, annotation, dataset construction, model training, quality evaluation, model versioning, and production monitoring. Defects receive bounding boxes or masks, while OK images receive empty label files. The dataset is split into training, validation, and test sets before training and evaluation, and each split is recorded in the dataset registry to prevent silent leakage or uncontrolled reuse.

The pipeline design uses dataset and model registries as control points. The dataset registry stores dataset versions, file manifests, split definitions, label statuses, and audit-log links. The model registry stores model versions, training parameters, evaluation metrics, model artefacts, and the dataset version used for training. This closes the gap between notebook experimentation and production governance because each deployed artefact can be traced back to a dataset and a training decision.

Quality gates are used before a model is promoted. They include standard detection measures, such as mean average precision at an Intersection-over-Union threshold of 0.50 (mAP@50), precision, and recall, as well as operational gates. The most important operational gate is the false-positive rate on OK samples. A model that detects many defects on clean products may look acceptable in a defect-heavy validation set but still be unusable in production. The completed MLOps flow and its corresponding governance outputs are summarized in Table 2.

**Table 2.** Completed MLOps flow for visual quality control.

| Stage                | Implemented Function                      | Governance Output                |
|----------------------|-------------------------------------------|----------------------------------|
| Data acquisition     | Store images/videos with metadata.        | Source metadata.                 |
| Frame selection      | Select stable frames.                     | Frame manifest.                  |
| ROI definition       | Define product area.                      | ROI metadata.                    |
| Annotation           | Add defect labels; OK labels stay empty.  | Label provenance.                |
| Dataset construction | Create train/validation/test splits.      | Dataset manifest.                |
| Training             | Train detector with recorded settings.    | Model artefact and training log. |
| Evaluation           | Calculate detection metrics and FP-on-OK. | Quality-gate report.             |
| Versioning           | Register dataset and model.               | Registry entries.                |
| Monitoring           | Track false alarms, drift, and latency.   | Monitoring log.                  |

#### 2.4. Human-in-the-Loop Annotation and Dataset Governance

Human review is built into the pipeline because defect labels are not merely technical annotations; they reflect quality rules. The pipeline therefore separates automated assistance from final accountability. Automated components can suggest boxes, propose labels, or prioritize uncertain samples, while human reviewers remain responsible for confirming ambiguous cases and quality-rule changes.

The annotation loop begins with a seed set of labelled data and expands through model-assisted review. The system records whether a label was manually drawn, accepted from a suggestion, corrected, or generated synthetically. This distinction is important because labels have different reliability levels and may require different review policies. Human-in-the-loop design also supports operator autonomy, which is central to Industry 5.0's framing of human-centric production [19].

Dataset governance prevents silent changes. The registry stores the manifest of images and labels, the training/validation/test split, the defect taxonomy, the ROI definition, the source of each label, and links to the audit logs. Data validation and monitoring are essential in production ML because invalid or shifting data can cause outages and model degradation [20].

#### 2.5. Synthetic Defect Generation Module

The NanoBanana augmentation module was implemented and executed through the synthetic generation and audit-logging stage as a notebook-level MLOps extension. Its purpose is to support defect classes that are rare, expensive to collect, or underrepresented in early deployment. The module uses OK images as clean surfaces and injects synthetic defects into the ROI. When the generated image contains a valid defect region, the corresponding label and bounding box are created automatically. Each generated sample is recorded in an audit log linking the source image, defect type, prompt or generation configuration, and output file. Here, NanoBanana denotes the Google Gemini 2.5 Flash Image accessed through the Gemini API (model identifier gemini-2.5-flash-image) [21].

The Task 1a notebook used the MVTec AD metal\_nut category [18]. The target classes were OK, SCRATCH, and STAIN. OK images were represented with empty labels, while SCRATCH and STAIN labels were produced from available defect masks. The notebook created a 60/20/20 train/validation/test split. It then profiled real training defects only, avoiding leakage from validation and test samples. The processed dataset contained 162 training images, 56 validation images, and 59 test images. The training split included 136 OK images, 13 SCRATCH images, and 13 STAIN images.

The augmentation procedure loaded defect profiles for scratch and colour/stain defects. The scratch profile was treated as linear, whereas the stain profile was treated as circular or blob-like. The generator created synthetic defects from OK images using prompts

and parameters derived from the real training defect profiles. The notebook generated synthetic SCRATCH and STAIN samples in verification mode, saved the resulting images and labels, and wrote a synthetic generation summary and audit log. For the recorded run, each request paired one OK training ROI with a defect-specific prompt. The model identifier, prompt or generation configuration, source file, and output file were stored in the generation-audit log. No fixed image seeds were recorded, so exact pixel-level replay has not been claimed.

The notebook defines the controlled comparison between real-only training and real-plus-synthetic training. YOLOv8n is used as the baseline detector, with fixed hyperparameters and a random seed [22,23]. The key methodological control is that the intended experimental difference is the training data and not the detector family, a random seed, or the evaluation split. In this manuscript, however, NanoBanana is reported primarily as an auditable generation and data-governance module. Any performance uplift from synthetic augmentation requires confirmation on held-out real defects and, preferably, repeated runs. The executed stages included dataset preparation, split construction, training-only defect profiling, synthetic generation, label serialization, and audit logging. Within this manuscript, Task 1a is therefore scoped as a data-governance and generation-audit experiment. Detector-level evaluation of the augmentation effect is defined by the protocol in Section 2.7 and is reserved for a separate future validation step with persistent artefact logging.

## 2.6. Machine Unlearning Module

The machine unlearning module was implemented and evaluated as a governed notebook-level operation. Its purpose was to support controlled removal of a defect class or data subset without always retraining from scratch. Relevant triggers included data recall, privacy or contractual restrictions, incorrect labels, changed defect taxonomy, and a quality-policy decision to stop detecting a class. Machine unlearning has been proposed as a way to remove the influence of selected data from trained systems [24], and SISA-style approaches show how restricting data influence can reduce deletion overhead [25].

The notebook implements approximate unlearning through restricted fine-tuning rather than certified deletion. This distinction is essential. The approach is practical and useful for production adaptation, but it is not a cryptographic guarantee that all influence has been removed. For formal deletion requirements, certified or legally audited methods would still be needed.

The workflow compares three models: `model_full`, trained on all classes; `model_retrained`, trained from scratch without the forgotten class and used as the reference for correct post-removal behaviour; and `model_unlearned`, produced by fine-tuning `model_full` on the restricted dataset. The restricted dataset is built by removing labels belonging to the forgotten class, dropping images that contain only the forgotten class, and remapping the retained class indices. The notebook also creates a forget set to test whether the model still detects the forgotten class. For acceptance, `model_unlearned` should be interpreted against `model_retrained` rather than only against `model_full`.

The executed unlearning scenario removed STAIN in a separate Task 1b stress-test corpus. The original Task 1b training partition contained 1344 images; after removing STAIN-only images, the restricted dataset kept 1025 training images and dropped 319. The original validation and test partitions contained 108 images each; after removal, each retained 86 images and dropped 22 STAIN-only images. In this corpus, OK remains the no-defect state; STAIN is the forgotten defect label; and the retained defect labels are SCRATCH, DISCOLORATION, and MISSING\_COMPONENT. One recorded artefact reports zero mAP@50 for the forgotten STAIN class with FP-on-OK values of 0.7463 for

model\_full and 0.0299 for model\_unlearned. The FP-on-OK value for model\_retrained was not retained in the available artefact; therefore, the full-versus-unlearned reduction is reported as diagnostic implementation evidence and not as a claim that approximate unlearning outperforms full retraining. For reproducibility, the source is identified as a project-internal AIMS 5.0 visual inspection stress-test corpus used for Task 1b rather than as a public benchmark. The frozen manifest, split lists, and class map encode the exact image sets and STAIN-only removal criteria, enabling independent verification of the experimental conditions under a data-use agreement.

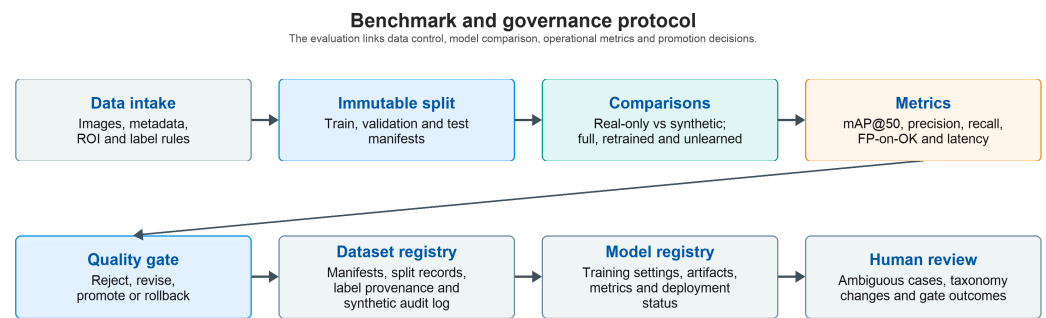
### 2.7. Benchmark Design and Evaluation Protocol

The benchmark design addresses the distinction between a platform blueprint and a research article. The evaluation protocol treats the visual quality-control service as an operational system with measurable data outputs, model outputs, and governance outputs (Table 3). The protocol therefore contains both conventional detector metrics and production-facing controls that determine whether a model is suitable for promotion. Operationally, data outputs refer to manifests and ROI records; model outputs refer to metrics and comparison reports; and governance outputs refer to registry entries and gate decisions.

**Table 3.** Benchmark design and quality-gate protocol.

| Evaluation Layer | Research Question                                 | Measurement or Gate                                                                                                                      | Evidence Generated                                                                               |
|------------------|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| Dataset          | Is the inspection data usable without leakage?    | Versioned manifests, split records, ROI definitions, empty OK labels, and label provenance.                                              | Dataset registry entry and audit log.                                                            |
| Detector         | Does the model detect localized defects?          | mAP@50, precision, recall, and class-level performance on held-out data.                                                                 | Evaluation report linked to model version.                                                       |
| Operations       | Would the model be tolerable in production?       | False-positive-on-OK rate, latency, review burden, and stability under monitored conditions.                                             | Quality-gate report and monitoring record.                                                       |
| Adaptation       | Can rare defects or removal requests be governed? | Auditable synthetic generation; full, retrained, and unlearned model comparison, with the retrained model as the post-removal reference. | Notebook artefacts, synthetic audit log, unlearning report, and explicit baseline-status record. |
| Promotion        | Can the result be safely released or rolled back? | Registry completeness, dataset–model traceability, threshold pass/fail decision, and rollback path.                                      | Model registry status and deployment decision.                                                   |

The primary dataset-level controls are immutable train/validation/test manifests, explicit OK images with empty labels, label-provenance fields, and the prevention of synthetic data leakage into validation or test partitions. The primary model-level controls are mAP@50, class-level recall, precision, and FP-on-OK rate. The primary operational controls are latency, review burden, monitoring stability, registry completeness, and rollback readiness. Together, these controls make the benchmark auditable and ensure that the claims remain limited to the evidence produced by the completed notebooks and demonstrations (Figure 3).



**Figure 3.** Benchmark and governance protocol. The figure shows the forward flow from data intake to the immutable split, controlled comparisons, operational metrics, and registry-backed quality-gate decisions. It also indicates which artefacts must exist before promotion or rollback.

### 3. Results

#### 3.1. Implemented OAP Use Cases

The battery carbon-footprint tool demonstrates a complete ontology-driven OAP workflow. Users can view the ontology and upload battery data as a TTL knowledge graph. During upload, the graph is validated against the ontology. If validation succeeds, the data are stored in Neo4j. The upload requires only the TTL file because key information, such as battery type and producer, is encoded in the graph.

The platform retrieves the stored knowledge graph from Neo4j and visualizes it. The graph includes entities such as the producer, product, carbon footprint, and emissions. The carbon-footprint entity contains fourteen emissions, grouped into Scope 1, Scope 2, and Scope 3 categories. The tool calculates the total carbon footprint across all fourteen emissions in kilograms and then provides scope-level summaries and detailed charts.

The same tool can also implement comparisons. A user can select a baseline battery and compare it against other batteries. The platform calculates whether the difference represents carbon avoidance, carbon burden, or a neutral impact. Download functions export the knowledge graph in a TTL or JSON format, and duplicate-upload detection prevents the same battery from being added twice without deliberate deletion and re-upload.

The production planning tool demonstrates a second OAP use case focused on manufacturing flow. It uses an ontology for wafer-fab data. The wafer-fab knowledge graph is uploaded in an Excel format, validated, and then stored in Neo4j. The graph view shows relationships between lots, parts, routes, steps, stations, and tools. The tool supports export into AnyLogic and AutoSched formats, which shows that validated production knowledge can be transformed into simulation or scheduling formats.

The production planning tool also supports lot-order monitoring. Users upload a lot-order text file, which is then validated and included in the database. The monitoring view displays the lots, parts, and step order, distinguishing between planned lots, work-in-progress lots, and finished lots. Through the graphical interface or API calls, the user can increase the step order so that lots move along their route. Monitoring outputs can be downloaded as text files in a zip archive, and plan comparisons are supported.

#### 3.2. Arrowhead Service Discovery and Authorization

The OAP integration with Eclipse Arrowhead demonstrates dynamic service discovery and governed access. Instead of hard-coding the URL of a service, an external consumer asks the Arrowhead orchestrator for the required service at runtime. The Arrowhead architecture uses service registry, orchestrator, and authorization systems [17].

In the implemented demonstration, AIMS/OAP backend services register automatically on startup. Registered services include health models, ontologies, monitoring and batteries. Each service has a URI; for example, the AIMS battery service maps to the

/batteries endpoint. A test consumer is registered with a name, address, and port. Authorization rules are then configured so that the consumer is permitted to access the selected AIMS services.

The consumer sends a request to the orchestrator asking for the battery service. The orchestrator checks the service registry to find the provider and checks the authorization to confirm that the requester is allowed to access it. The response returns the provider, service URI, and interface. A Python 3.11 test consumer script automates the full flow by querying the orchestrator, extracting the endpoint, and calling the discovered service.

The demonstration also validates the negative case. An unauthorized consumer can be registered without access rules, but when it requests the AIMS battery service, the response is empty. This confirms that discovery and access are not open by default.

### 3.3. Evidence-Supported Implementation Status (Table 4)

This section summarizes the evidence basis for each implemented or notebook-level component. Table 4 separates the demonstrated platform functions from the visual-inspection adaptation modules and states the main qualification that limits each claim.

**Table 4.** Evidence-supported implementation status.

| Component                     | Evidence-Supported Status                                                                                                                                          | Important Qualification                                                                                                                                   |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Battery carbon-footprint tool | Implemented OAP tool with ontology viewing, TTL upload, Neo4j storage, graph visualization, footprint calculation, comparison, export, and duplicate detection.    | Evidence is based on the supplied demonstration transcript.                                                                                               |
| Production planning tool      | Implemented OAP tool with wafer-fab ontology, Excel upload, Neo4j graph exploration, AnyLogic/ AutoSched export, lot monitoring, API control, and plan comparison. | Evidence is based on the supplied demonstration transcript.                                                                                               |
| Arrowhead integration         | Implemented discovery flow with service registration, orchestrator query, authorization check, endpoint return, successful access, and unauthorized denial.        | Evidence is based on the supplied demonstration transcript and the Arrowhead framework reference [17].                                                    |
| NanoBanana augmentation       | Implemented as a notebook-level optional module with synthetic defect generation and audit logging.                                                                | Evidence supports generation governance and leakage control; no general performance uplift is claimed without held-out real-defect validation.            |
| Machine unlearning            | Implemented as a notebook-level approximate unlearning module with restricted dataset, forget set, model comparison, and audit outputs.                            | The Task 1b stress test uses a separate corpus and taxonomy; model_retrained is the required reference, and the method does not certify deletion [24,25]. |

### 3.4. Monitoring, Registries, and Quality Gates

Monitoring closes the MLOps loop. A visual inspection model can degrade after deployment even when it has passed initial tests. Drift may appear due to changed lighting, new material textures, altered camera positions, different product geometries, or new defect types. The monitoring layer therefore tracks data quality, model outputs, confidence patterns, prediction rates, defect rates, false-positive-on-OK rates, latency, errors, and human review loads.

Quality gates are used at three points. Dataset gates check whether the data split, label formats, ROI definitions, and audit logs are complete. Model gates check detection metrics, false-positive-on-OK rates, and cross-condition evaluation. Deployment gates check whether the artefact is registered, linked to the dataset version, and has rollback information. This follows the production ML guidance that data, model, infrastructure, and monitoring tests should be considered together [2,26].

The dataset registry connects image manifests, label files, split definitions, defect taxonomy, ROI metadata, synthetic sample metadata, and unlearning logs. The model registry connects model artefacts, training configuration, metrics, comparison reports, deployment statuses, and parent dataset version. Together, they enable controlled promotion and rollback.

### 3.5. Quantitative Benchmark and Operational Evidence

The results separate qualitative implementation evidence from quantitative benchmark evidence and explicitly audits the consistency of dataset identity, class taxonomy, and model comparisons. The available measurements are intentionally reported at the level supported by the completed notebooks and demonstration artefacts. This improves scientific transparency while avoiding unsupported extrapolation from small or single-run experiments (Table 5). Within the benchmark protocol, the executed stages included dataset construction and split generation for Tasks 1a and 1b; synthetic generation with audit logging for Task 1a; and full-model and approximate-unlearning training with forgotten class and available FP-on-OK evaluations for Task 1b. This keeps the evidence boundary visible at the point where results are reported.

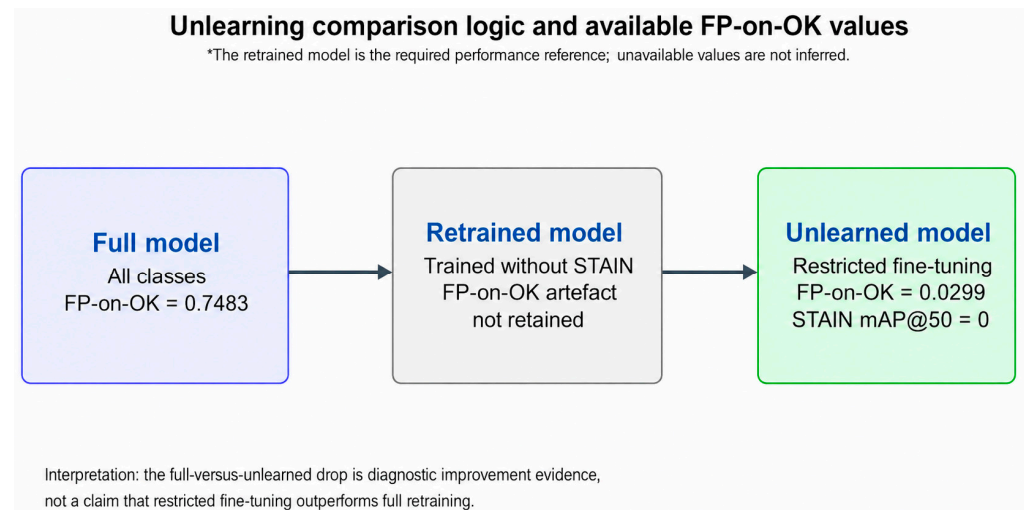
**Table 5.** Quantitative evidence extracted from the completed notebook-level modules. The table keeps Task 1a and Task 1b separate and reports only the retained evidence.

| Evidence Item                 | Reported Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Interpretation for the Benchmark                                                                                                                                              |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Task 1a’s dataset and split   | MVTec AD metal_nut; 162 training, 56 validation, and 59 test images. Task 1a states are OK, SCRATCH, and STAIN; the training split contains 136 OK, 13 SCRATCH, and 13 STAIN images.                                                                                                                                                                                                                                                                                                        | This is a small public benchmark used for augmentation protocol and class-imbalance analysis, not a broad production validation.                                              |
| Synthetic augmentation status | NanoBanana generated auditable synthetic SCRATCH and STAIN verification samples from OK training images; synthetic samples are excluded from validation and test partitions. Task 1a is scoped here to synthetic generation governance and audit logging. Detector-level augmentation outputs (mAP@50, precision, recall, class-level results, and FP-on-OK) were not retained in the available evidence package and are reserved for future re-execution with persistent artefact logging. | Evidence supports governed generation, label/audit logging, and leakage control; no general performance uplift is claimed without additional held-out real-defect validation. |
| Task 1b’s corpus identity     | Separate unlearning stress-test corpus. Original training partition—1344 images; retained after STAIN-only removal—1025; and dropped—319. Original validation/test partitions—108 each; retained—86 each; and dropped—22 each. Source: project-internal AIMS 5.0 visual inspection stress-test corpus for Task 1b; the corpus is not publicly released.                                                                                                                                     | The corpus is not a Task 1a MVTec metal_nut split; this separation prevents cross-task confusion.                                                                             |

Table 5. Cont.

| Evidence Item                | Reported Value                                                                                                                                                            | Interpretation for the Benchmark                                                                                                            |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Task 1b taxonomy             | OK is the no-defect state. STAIN is the forgotten label. Retained defect labels are SCRATCH, DISCOLORATION, and MISSING_COMPONENT.                                        | Class names are task-specific and are reported independently from the Task 1a OK/SCRATCH/STAIN taxonomy.                                    |
| Model comparison requirement | The model_full is trained on all classes; model_retrained is trained from scratch without STAIN; and model_unlearned is produced by restricted fine-tuning without STAIN. | The model_retrained is the correct post-removal reference; full-versus-unlearned values are diagnostic only.                                |
| Available unlearning values  | The model_unlearned achieved zero mAP@50 for forgotten STAIN with FP-on-OK = 0.0299; model_full obtained FP-on-OK = 0.7463.                                               | The available artefact did not retain FP-on-OK for model_retrained, so the manuscript does not claim superiority over full retraining.      |
| Variability status           | Values are from one recorded notebook run with fixed settings.                                                                                                            | Results are reported as implementation evidence; repeated-run variability is identified for future validation before production acceptance. |

Figure 4 links the available FP-on-OK values to the required unlearning comparison logic and makes the interpretation deliberately narrow. The manuscript avoids unsupported claims; therefore, augmentation is presented as a governed generation module, and unlearning is presented as an approximate adaptation workflow whose retained-class and false-positive behaviour should be checked against the retrained reference before production promotion.



**Figure 4.** Unlearning comparison logic and available FP-on-OK values. The retrained model is identified as the required post-removal reference; unavailable retrained FP-on-OK values are not inferred. Only the recorded values are shown, and the missing retrained FP-on-OK value is treated as unavailable.

#### 4. Discussion

The main contribution of this work is a governed service architecture for industrial visual inspections. The OAP evidence demonstrates two working semantic tools and a service-discovery integration. The battery carbon-footprint tool shows ontology validation, TTL knowledge-graph upload, Neo4j storage, graph visualization, emissions calculation, comparison, export, and duplicate detection. The production planning tool shows ontology-

guided wafer-fab upload, Neo4j graph exploration, simulation-format export, lot-order monitoring, step progression, API control, plan comparison, and deletion.

These OAP capabilities matter because visual quality control needs the same platform functions: semantic consistency, governed upload, graph traceability, API access, and monitoring. The pipeline described in this manuscript is therefore more than an ML training loop; it is an operational lifecycle that connects datasets, labels, models, audit records, and service integrations.

The NanoBanana and unlearning notebooks extend the platform pattern into two practical adaptation scenarios. Synthetic augmentation addresses insufficient real defects while preserving auditability of generated samples, whereas the present evidence supports generation governance rather than a general accuracy improvement claim. Approximate unlearning addresses changing inspection rules and governed class removal, but the appropriate comparator is the retrained model. Both modules are framed cautiously: the unlearning module was evaluated in a notebook-level workflow, whereas Task 1a synthetic augmentation was executed through the generation and audit-logging stage. Neither module is presented as a fully deployed OAP service or a formally certified data-deletion mechanism.

From an Industry 5.0 perspective, the system is human-centric because the human reviewer remains responsible for ambiguous cases and quality rules. It can support sustainability because better inspection and controlled augmentation may reduce wasteful over-collection and repeated manual effort. It supports resilience because the platform includes monitoring, versioning, rollback, and modular adaptation when conditions change [19].

### *Limitations*

Several limitations remain. First, the OAP demonstrations verify the platform's capabilities for battery footprint, production planning, and Arrowhead integration, but the current evidence does not yet show the visual quality-control service expressed as a live OAP or Arrowhead service. The manuscript therefore describes the visual inspection lifecycle as an OAP-aligned service blueprint with implemented notebook-level modules and not as a fully deployed interface.

Second, NanoBanana augmentation was executed in a controlled notebook setting through the synthetic generation and audit-logging stage, with limited verification mode generation in the retained evidence package. This supports the module design, auditability, and leakage-control methodology, but it does not prove broad production gains across many product types or defect categories. Synthetic defects must be validated on held-out real defects before promotion; therefore, synthetic augmentation is not treated as a proven performance improvement. Within this manuscript, Task 1a is scoped as a data-governance and generation-audit experiment; as such, detector-level augmentation evaluation (mAP@50, precision, recall, class-level results and FP-on-OK) is reserved for future re-execution with persistent artefact logging.

Third, the unlearning workflow is approximate and reported as separate from Task 1b's stress-test corpus rather than from the small Task 1a MVTec split. Fine-tuning on a restricted dataset can reduce forgotten-class detection and may save time compared with full retraining, but it is not a certified guarantee that all influences of the removed data has been eliminated. The retrained model remains the correct post-removal reference. Because the available artefact did not retain FP-on-OK for model\_retrained, the reported full-versus-unlearned FP-on-OK change is diagnostic only. If the use case requires legally enforceable deletion, then stronger certified unlearning or full retraining may be required.

Fourth, the manuscript does not claim to be a universal defect detector. The Task 1a split is intentionally small and imbalanced, and the Task 1b stress test has a different

taxonomy. The defect taxonomies, ROI definitions, acceptable false-positive thresholds, and quality gates must therefore be adapted for each production line.

These limitations are also design constraints for future work. A full industrial validation should add plant-specific image streams, cross-shift and cross-lighting tests, repeated training runs or confidence intervals, longer monitoring periods, operator workload measurements, and service-level deployment through OAP and Arrowhead architectures. The present manuscript provides the governed blueprint, benchmark protocol, and initial implementation evidence needed to support this next validation stage.

## 5. Conclusions

This article presents an evidence-grounded blueprint for visual quality control as a governed service within OAPs. The OAP context is supported by the implementation of battery carbon-footprint and production planning tools, both of which are based on ontology-driven validation and Neo4j-backed knowledge graphs. Arrowhead integration demonstrates dynamic discovery and authorization, removing the need for hard-coded service endpoints and supporting Industry 4.0-style interoperability.

The visual quality-control workflow is structured as a complete MLOps lifecycle: acquisition, frame selection, ROI definition, annotation, dataset construction, training, evaluation, versioning, and monitoring. Dataset and model registries, quality gates, and audit logs make the lifecycle traceable. The benchmark design includes leakage-safe split controls, task-specific taxonomy controls, explicit model-comparison requirements, and FP-on-OK as a production-critical metric.

Two optional modules extend the pipeline. NanoBanana synthetic defect generation supports rare or imbalanced defect classes by generating auditable synthetic samples from OK images, though the current evidence is limited to governed generation and auditability. Approximate machine unlearning supports governed class or data removal through restricted fine-tuning and comparison requirements against full retraining. Overall, the blueprint supports human-in-the-loop decision-making, semantic traceability, modular adaptation, controlled promotion, and Industry 5.0-aligned industrial AI governance. For Industry 5.0, the central accountability mechanism is the explicit human review of the dataset, model, and promotion gates, not hidden automation.

**Author Contributions:** Conceptualization, K.W. and J.N.; methodology, K.W., M.S.T. and P.K.-O.; software, J.N., M.S.T., P.K.-O. and M.K.; validation, K.W., J.N., M.S.T., P.K.-O. and M.K.; formal analysis, K.W. and M.S.T.; investigation, K.W., J.N. and P.K.-O.; resources, K.W. and J.N.; data curation, M.S.T., P.K.-O. and M.K.; writing—original draft preparation, K.W.; writing—review and editing, K.W., J.N., M.S.T., P.K.-O. and M.K.; visualization, M.S.T., P.K.-O. and M.K.; supervision, K.W. and J.N.; project administration, K.W.; funding acquisition, K.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was carried out as part of the AIMS 5.0 project. The AIMS 5.0 project is supported by the Chips Joint Undertaking and its members, including top-up funding from the National Funding Authorities in the participating countries under the grant agreement no. 101112089.

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

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The public benchmark data used for notebook-level experiments are available from the MVTec AD dataset [18]. OAP demonstration transcripts, processed manifests, audit logs, and notebook artefacts can be made available as supplementary implementation evidence where licencing and project-confidentiality constraints permit. The separate Task 1b corpus is project-internal and cannot be redistributed publicly; its frozen manifests, split lists, class maps, and file hashes can be provided for confidential review or under an appropriate data-use agreement.

**Acknowledgments:** The authors acknowledge the technical and administrative support received during the AIMS 5.0 project and the implementation work underlying the OAP demonstrations, visual quality-control notebooks and governance artefacts described in this manuscript.

**Conflicts of Interest:** Author Krzysztof Wołk was employed by the company DAC.next Sp. z o.o. Authors Jacek Niklewski, Marek S. Tatała, Piotr Kopa-Ostrowski and Michał Kopczyński were employed by the company DAC.Digital S.A. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## Abbreviations

The following abbreviations are used in this manuscript:

|       |                                   |
|-------|-----------------------------------|
| AI    | Artificial Intelligence           |
| FP    | False Positive                    |
| ML    | Machine Learning                  |
| MLOps | Machine Learning Operations       |
| OAP   | Open Access Platform              |
| ROI   | Region of Interest                |
| TTL   | Terse RDF Triple Language         |
| API   | Application Programming Interface |
| HITL  | Human-in-the-Loop                 |
| IoU   | Intersection-over-Union           |
| mAP   | Mean Average Precision            |
| VQC   | Visual Quality Control            |

## References

1. Sculley, D.; Holt, G.; Golovin, D.; Davydov, E.; Phillips, T.; Ebner, D.; Chaudhary, V.; Young, M.; Crespo, J.-F.; Dennison, D. Hidden Technical Debt in Machine Learning Systems. In Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015), Montreal, QC, Canada, 7–12 December 2015; pp. 2503–2511.
2. Breck, E.; Cai, S.; Nielsen, E.; Salib, M.; Sculley, D. The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 1123–1132. [\[CrossRef\]](#)
3. Quiñero-Candela, J.; Sugiyama, M.; Schwaighofer, A.; Lawrence, N.D. (Eds.) *Dataset Shift in Machine Learning*; MIT Press: Cambridge, MA, USA, 2009.
4. Recht, B.; Roelofs, R.; Schmidt, L.; Shankar, V. Do ImageNet Classifiers Generalize to ImageNet? In *Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019*; Proceedings of Machine Learning Research: Cambridge, MA, USA, 2019; Volume 97, pp. 5389–5400.
5. Amershi, S.; Cakmak, M.; Knox, W.B.; Kulesza, T. Power to the People: The Role of Humans in Interactive Machine Learning. *AI Mag.* **2014**, *35*, 105–120. [\[CrossRef\]](#)
6. Mosqueira-Rey, E.; Hernández-Pereira, E.; Alonso-Ríos, D.; Bobes-Bascarán, J.; Fernández-Leal, Á. Human-in-the-Loop Machine Learning: A State of the Art. *Artif. Intell. Rev.* **2023**, *56*, 3005–3054. [\[CrossRef\]](#)
7. Settles, B. *Active Learning Literature Survey*; Computer Sciences Technical Report 1648; University of Wisconsin-Madison: Madison, WI, USA, 2009.
8. Sohn, K.; Berthelot, D.; Li, C.-L.; Zhang, Z.; Carlini, N.; Cubuk, E.D.; Kurakin, A.; Zhang, H.; Raffel, C. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020; pp. 596–608.
9. Kreuzberger, D.; Kühl, N.; Hirsch, S. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access* **2023**, *11*, 31866–31879. [\[CrossRef\]](#)
10. Kumara, I.; Arts, R.; Ferreira, R.C.; Di Nucci, D.; Kazman, R.; Tamburri, D.A.; van den Heuvel, W.-J. MLOps in Practice: Requirements and a Reference Architecture from Industry. In *Software Architecture: 19th European Conference, ECSA 2025, Limassol, Cyprus, 15–19 September 2025, Proceedings*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2026; Volume 15929, pp. 20–37. [\[CrossRef\]](#)
11. Rajashekarappa, M.; Turanoglu Bekar, E.; Karlsson, A.; Bokrantz, J.; Subramaniyan, M.; Skoogh, A. Industrial MLOps: A Systematic Review of Architectures and Implementation Challenges. *Prod. Manuf. Res.* **2026**, *14*, 2658878. [\[CrossRef\]](#)

12. Lim, J.; Lee, H.; Won, Y.; Yeon, H. MLOp Lifecycle Scheme for Vision-Based Inspection Process in Manufacturing. In Proceedings of the 2019 USENIX Conference on Operational Machine Learning (OpML 19), Santa Clara, CA, USA, 20 May 2019; pp. 9–11.
13. Baylor, D.; Breck, E.; Cheng, H.-T.; Fiedel, N.; Foo, C.Y.; Haque, Z.; Haykal, S.; Ispir, M.; Jain, V.; Koc, L.; et al. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17), Halifax, NS, Canada, 13–17 August 2017; pp. 1387–1395. [\[CrossRef\]](#)
14. Steidl, M.; Felderer, M.; Ramler, R. The Pipeline for the Continuous Development of Artificial Intelligence Models—Current State of Research and Practice. *J. Syst. Softw.* **2023**, *199*, 111615. [\[CrossRef\]](#)
15. Delsing, J.; Varga, P.; Ferreira, L.; Albano, M.; Pereira, P.P.; Eliasson, J.; Carlsson, O.; Derhamy, H. The Arrowhead Framework Architecture. In *IoT Automation: Arrowhead Framework*; Delsing, J., Ed.; CRC Press: Boca Raton, FL, USA, 2017; pp. 43–88. [\[CrossRef\]](#)
16. Kejriwal, M. Knowledge Graphs: A Practical Review of the Research Landscape. *Information* **2022**, *13*, 161. [\[CrossRef\]](#)
17. Eclipse Foundation. Eclipse Arrowhead. Available online: <https://projects.eclipse.org/projects/iot.arrowhead> (accessed on 1 May 2026).
18. Bergmann, P.; Fauser, M.; Sattlegger, D.; Steger, C. MVTec AD-A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 9584–9592. [\[CrossRef\]](#)
19. European Commission; Directorate-General for Research and Innovation. *Industry 5.0: Towards a Sustainable, Human-Centric and Resilient European Industry*; Publications Office of the European Union: Luxembourg, 2021. [\[CrossRef\]](#) [\[PubMed\]](#)
20. Breck, E.; Polyzotis, N.; Roy, S.; Whang, S.E.; Zinkevich, M. Data Validation for Machine Learning. In Proceedings of the MLSys 2019, Stanford, CA, USA, 31 March–2 April 2019.
21. Google. Gemini 2.5 Flash Image (Nano Banana). Available online: <https://ai.google.dev/gemini-api/docs/models/gemini-2.5-flash-image> (accessed on 16 July 2026).
22. Ultralytics. YOLOv8 Documentation. Available online: <https://docs.ultralytics.com/models/yolov8/> (accessed on 1 May 2026).
23. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [\[CrossRef\]](#)
24. Cao, Y.; Yang, J. Towards Making Systems Forget with Machine Unlearning. In Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–21 May 2015; pp. 463–480. [\[CrossRef\]](#)
25. Bourtole, L.; Chandrasekaran, V.; Choquette-Choo, C.A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; Papernot, N. Machine Unlearning. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 24–27 May 2021; pp. 141–159. [\[CrossRef\]](#)
26. Schröder, T.; Schulz, M.; Christ, M.; Lins, D.; Kraus, M.; Zschech, P. Monitoring Machine Learning Models: A Categorization of Challenges and Methods. *Data Knowl. Eng.* **2022**, *139*, 102028. [\[CrossRef\]](#)

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.